SAM E70

Atmel | SMART ARM-based Flash MCU

DATASHEET

Introduction

Atmel® | SMART SAM E70 is a high-performance Flash microcontroller (MCU) based on the 32-bit ARM® Cortex®-M7 RISC (5.04 CoreMark/MHz) processor with floating point unit (FPU). The device operates at a maximum speed of 300 MHz, features up to 2048 Kbytes of Flash, dual 16 Kbytes of cache memory, up to 384 Kbytes of SRAM and is available in 64-, 100- and 144-pin packages.

The Atmel | SMART SAM E70 offers an extensive peripheral set, including Ethernet 10/100, dual CAN-FD, High-speed USB Host and Device plus PHY, up to 8 UARTs, I2S, SD/MMC interface, a CMOS camera interface, system control and a 12-bit 2 Msps ADC, as well as high-performance crypto-processors AES, SHA and TRNG.

Features

- Core
  - ARM Cortex-M7 running at up to 300 MHz(1)
  - 16 Kbytes of ICache and 16 Kbytes of DCache with Error Code Correction (ECC)
  - Simple- and double-precision HW Floating Point Unit (FPU)
  - Memory Protection Unit (MPU) with 16 zones
  - DSP Instructions, Thumb®-2 Instruction Set
  - Embedded Trace Module (ETM) with instruction trace stream, including Trace Port Interface Unit (TPIU)
- Memories
  - Up to 2048 Kbytes embedded Flash with unique identifier and user signature for user-defined data
  - Up to 384 Kbytes embedded Multi-port SRAM
  - Tightly Coupled Memory (TCM) interface with four configurations (disabled, 2 x 32 Kbytes, 2 x 64 Kbytes, 2 x 128 Kbytes)
  - 16 Kbytes ROM with embedded Boot Loader routines (UART0, USB) and IAP routines
  - 16-bit Static Memory Controller (SMC) with support for SRAM, PSRAM, LCD module, NOR and NAND Flash with on-the-fly scrambling
  - 16-bit SDRAM Controller (SDRAMC) interfacing up to 256 MB and with on-the-fly scrambling
- System
  - Embedded voltage regulator for single-supply operation
  - Power-on-Reset (POR), Brown-out Detector (BOD) and Dual Watchdog for safe operation

Downloaded from Arrow.com.

- Quartz or ceramic resonator oscillators: 3 to 20 MHz main oscillator with failure detection, 12 MHz or 16 MHz needed for USB operations. Optional low-power 32.768 kHz for RTC or device clock
- RTC with Gregorian calendar mode, waveform generation in low-power modes
- RTC counter calibration circuitry compensates for 32.768 kHz crystal frequency variations
- 32-bit low-power Real-time Timer (RTT)
- High-precision 4/8/12 MHz internal RC oscillator with 4 MHz default frequency for device startup. In-application trimming access for frequency adjustment. 8/12 MHz are factory-trimmed.
- 32.768 kHz crystal oscillator or embedded 32 kHz (typical) RC oscillator as source of low-power mode device clock (SLCK)
- One 500 MHz PLL for system clock, one 480 MHz PLL for USB high-speed operations
- Temperature Sensor
- One dual-port 24-channel central DMA Controller (XDMAC)

- Low-Power Features
  - Low-power Sleep, Wait and Backup modes, with typical power consumption down to 1.1 µA in Backup mode with RTC, RTT and wake-up logic enabled
  - Ultra-low-power RTC and RTT
  - 1 Kbyte of backup RAM (BRAM) with dedicated regulator
- Peripherals
  - One Ethernet MAC (GMAC) 10/100 Mbps in MII mode and RMII with dedicated DMA. IEEE1588 PTP frames and 802.3az Energy-efficiency support. Ethernet AVB support with IEEE802.1AS Time-stamping and IEEE802.1Qav credit-based traffic-shaping hardware support.
  - USB 2.0 Device/Mini Host High-speed (USBHS) at 480 Mbps, 4-Kbyte FIFO, up to 10 bidirectional endpoints, dedicated DMA
  - 12-bit ITU-R BT. 601/656 Image Sensor Interface (ISI)
  - Two master Controller Area Networks (MCAN) with Flexible Data Rate (CAN-FD) with SRAM-based mailboxes, time- and event-triggered transmission
  - Three USARTs. USART0/1/2 support LIN mode, ISO7816, IrDA®, RS-485, SPI, Manchester and Modem modes; USART1 supports LON mode.
  - Five 2-wire UARTs with SleepWalking support
  - Three Two-Wire Interfaces (TWIHS) (I^2C-compatible) with SleepWalking support
  - Quad I/O Serial Peripheral Interface (QSPI) interfacing up to 256 MB Flash and with eXecute-In-Place and on-the-fly scrambling
  - Two Serial Peripheral Interfaces (SPI)
  - One Serial Synchronous Controller (SSC) with I2S and TDM support
  - Two Inter-IC Sound Controllers (I2SC)
  - One High-speed Multimedia Card Interface (HSMCI) (SDIO/SD Card/e.MMC)
  - Four Three-Channel 16-bit Timer/Counters (TC) with Capture, Waveform, Compare and PWM modes, constant on time. Quadrature decoder logic and 2-bit Gray Up/Down Counter for stepper motor
  - Two 4-channel 16-bit PWMs with complementary outputs, Dead Time Generator and eight fault inputs per PWM for motor control, two external triggers to manage power factor correction (PFC), DC-DC and lighting control.
  - Two Analog Front-End Controllers (AFEC), each supporting up to 12 channels with differential input mode and programmable gain stage, allowing dual sample-and-hold at up to 2 Msps. Gain and offset error autotest feature.
  - One 2-channel 12-bit 1Msps-per-channel Digital-to-Analog Controller (DAC) with differential and oversampling modes
  - One Analog Comparator (ACC) with flexible input selection, selectable input hysteresis
- Cryptography
  - True Random Number Generator (TRNG)
  - AES: 256-, 192-, 128-bit Key Algorithm, Compliant with FIPS PUB-197 Specifications
  - Integrity Check Monitor (ICM). Supports Secure Hash Algorithm SHA1, SHA224 and SHA256.

Downloaded from Arrow.com.

- I/O
  - Up to 114 I/O lines with external interrupt capability (edge- or level-sensitivity), debouncing, glitch filtering and On-die Series Resistor Termination
  - Five Parallel Input/Output Controllers (PIO)
- Voltage
  - Single supply voltage from 1.7V to 3.6V
- Packages
  - LQFP144, 144-lead LQFP, 20 x 20 mm, pitch 0.5 mm
  - LFBGA144, 144-ball LFBGA, 10 x 10 mm, pitch 0.8 mm
  - UFBGA144, 144-ball UFBGA, 6 x 6 mm, pitch 0.4 mm(2)
  - LQFP100, 100-lead LQFP, 14 x 14 mm, pitch 0.5 mm
  - TFBGA100, 100-ball TFBGA, 9 x 9 mm, pitch 0.8 mm
  - LQFP64, 64-lead LQFP, 10 x 10 mm, pitch 0.5 mm

Notes: 1. 300 MHz is at [-40°C : +105°C], 1.2V or with the internal regulator.
2. Contact your local Atmel sales representative for availability.

Downloaded from Arrow.com.

1. Description

The Atmel | SMART SAM E70 devices are members of a family of Flash microcontrollers based on the high-performance 32-bit ARM Cortex-M7 processor with Floating Point Unit (FPU). These devices operate at up to 300 MHz and feature up to 2048 Kbytes of Flash and up to 384 Kbytes of multi-port SRAM.

The on-chip SRAM can be configured as Tightly Coupled Memory (TCM) or system memory. A multi-port access to the SRAM guarantees a minimum access latency.

The peripheral set includes an Ethernet MAC (GMAC) supporting AVB, IEEE1588, 802.1Qbb, 802.3az, 802.1AS and 802.1Qav, a high-speed USB Device port and a high-speed USB Host port sharing an embedded transceiver, an Image Sensor Interface (ISI), a high-speed Multimedia Card Interface (HSMCI) for SDIO/SD/e.MMC, an External Bus Interface (EBI) featuring an SDRAM Controller, and a Static Memory Controller providing connection to SRAM, PSRAM, NOR Flash, LCD module and NAND Flash. Additional peripherals include three Universal Synchronous Asynchronous Receiver Transmitters (USART), five Universal Asynchronous Receiver Transmitters (UART), three Two-wire Interfaces (TWI) supporting the I^2C protocol, one Quad I/O Serial Peripheral Interface (QSPI), two Serial Peripheral Interfaces (SPI), one Serial Synchronous Controller (SSC) supporting I2S and TDM protocols, two Inter-IC Sound Controllers (I2SC), as well as two enhanced Pulse Width Modulators (PWM), twelve general-purpose 16-bit timers with stepper motor and quadrature decoder logic support, two Controller Area Networks with Flexible Data Rate (CAN-FD), one ultra low-power Real-Time Timer (RTT), one ultra low-power Real-Time Clock (RTC), dual Analog Front-End (AFE) including a 12-bit Analog-to-Digital Converter (ADC), a Programmable Gain Amplifier (PGA), dual Sample-and-Hold and a digital averaging with up to 16-bit resolution, dual-channel 12-bit Digital-to-Analog Converter (DAC) and one Analog Comparator, as well as high-performance crypto-processors Advanced Encryption Standard (AES), Secure Hash Algorithm (SHA) and True Random Number Generator (TRNG).

The SAM E70 devices have three software-selectable low-power modes: Sleep, Wait and Backup. In Sleep mode, the processor is stopped while all other functions can be kept running. In Wait mode, all clocks and functions are stopped but some peripherals can be configured to wake up the system based on predefined conditions. This feature, called SleepWalking™, performs a partial asynchronous wake-up, thus allowing the processor to wake up only when needed. In Backup mode, RTT, RTC and wake-up logic are running. Optionally a 1-Kbyte low-power SRAM can be retained.

To optimize power consumption, the clock system has been designed to support different clock frequencies for selected peripherals. Moreover, the processor and bus clock frequency can be modified without affecting processing on, for example, the USB, U(S)ART, AFE and Timer Counter.

The SAM E70 devices also feature an event system that allows peripherals to receive, react to and send events in Active and Sleep modes without processor intervention.

The SAM E70 devices are high-performance general-purpose microcontrollers with a rich set of connectivity peripherals and large memory integration. This enables the SAM E70 to sustain a wide range of applications including consumer, industrial control, and PC peripherals.

SAM E70 devices operate from 1.7V to 3.6V and are pin-to-pin compatible with the SAM4E (100-pin and 144-pin versions), except for USB signals.

The Atmel application note “Migrating the SAM4E to SAM E70 Microcontroller” (reference 44034) is available on www.atmel.com to ease migration from SAM4E devices to SAM E70 devices.

Downloaded from Arrow.com.

Downloaded from Arrow.com.

2. Configuration Summary

The SAM E70 devices differ in memory size, package and features. Table 2-1 summarizes the different configurations.

Table 2-1. Configuration Summary

Feature	SAME70Q21	SAME70Q20	SAME70Q19	SAME70N21	SAME70N20	SAME70N19	SAME70J21	SAME70J20	SAME70J19
Flash (Kbytes)	2048	1024	512	2048	1024	512	2048	1024	512
Flash Page Size (bytes)	512								
Flash Pages	4096	2048	1024	4096	2048	1024	4096	2048	1024
Flash Lock Region Size (Kbytes)	16								
Flash Lock Bits	128	64	32	128	64	32	128	64	32
Multi-port SRAM (Kbytes)	384		256	384		256	384		256
Cache(I/D) (Kbytes)	16/16								
Package	LQFP144 LFBGA144 UFBGA144	LQFP144 LFBGA144 UFBGA144	LQFP144 LFBGA144 UFBGA144	LQFP100 TFBGA100	LQFP100 TFBGA100	LQFP100 TFBGA100	LQFP64	LQFP64	LQFP64
Number of PIOs	114			75			44		
External Bus Interface	16-bit data, 4 chip selects, 24-bit address			–			–		
SDRAM Interface	Yes			–			–		
Central DMA	24			24			24		
12-bit ADC	24 ch.(2)			10 ch.(2)			5 ch.(2)		
12-bit DAC	2 ch.			2 ch.			1 ch.		
Timer Counter Channels	12								
Timer Counter Channels I/O	36			9			3		
USART/UART	3/5(1)			3/5(1)			0/5		
QSPI	Yes			Yes			SPI mode only		
SPI0	Yes			Yes			No		

Downloaded from Arrow.com.

Table 2-1. Configuration Summary (Continued)

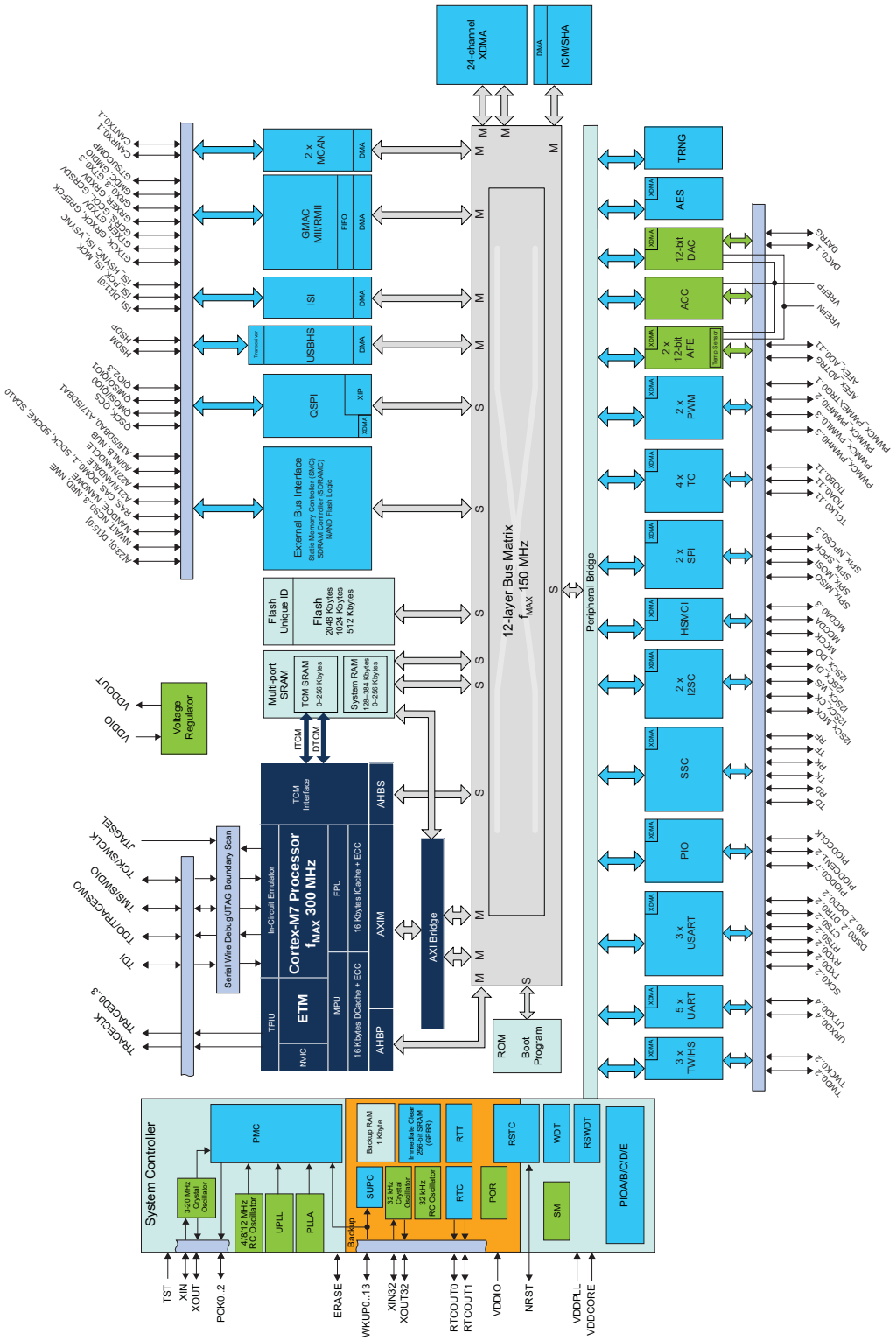
Feature	SAME70Q21	SAME70Q20	SAME70Q19	SAME70N21	SAME70N20	SAME70N19	SAME70J21	SAME70J20	SAME70J19
SPI1	Yes			No			No		
USART SPI	3(1)			3(1)			0		
TWIHS	3			3			2		
HSMCI	1 port 4 bits			1 port 4 bits			–		
CAN	2 ports			2 ports			1 port		
GMAC	MII, RMII			MII, RMII			RMII		
ISI	12-bit			12-bit			8-bit		
SSC	Yes			Yes			Yes		
I2SC	2			1			0		
USB	High-speed			High-speed			Full-speed		
Analog Comparator	Yes			Yes			Yes		
Embedded Trace Macrocell (ETM)	Yes			Yes			Yes		

Notes: 1. LON support on USART1 only.
2. One channel is reserved for internal temperature sensor.

3. Block Diagram

See Table 2-1 for detailed configurations of memory size, package and features of the SAM E70 devices.

Figure 3-1. SAM E70 144-pin Block Diagram



Downloaded from Arrow.com.

4. Signal Description

Table 4-1 gives details on signal names classified by peripheral.

Table 4-1. Signal Description List

Signal Name	Function	Type	Active Level	Voltage Reference	Comments
Power Supplies					
VDDIO	Peripherals I/O Lines Power Supply	Power	–	–	–
VDDIN	Voltage Regulator Input, ADC, DAC and Analog Comparator Power Supply(1)	Power	–	–	–
VDDOUT	Voltage Regulator Output	Power	–	–	–
VDDPLL	PLLA Power Supply	Power	–	–	–
VDDPLLUSB	USB PLL and Oscillator Power Supply	Power	–	–	–
VDDCORE	Powers the core, the embedded memories and the peripherals	Power	–	–	–
GND, GNDPLL, GNDPLLUSB, GNDANA, GNDUTMI	Ground	Ground	–	–	–
VDDUTMII	USB Transceiver Power Supply	Power	–	–	–
VDDUTMIC	USB Core Power Supply	Power	–	–	–
GNDUTMI	USB Ground	Ground	–	–	–
Clocks, Oscillators and PLLs					
XIN	Main Oscillator Input	Input	–	VDDIO	–
XOUT	Main Oscillator Output	Output	–		–
XIN32	Slow Clock Oscillator Input	Input	–		–
XOUT32	Slow Clock Oscillator Output	Output	–		–
PCK0–PCK2	Programmable Clock Output	Output	–		PCK3 is TRACE clock PCK4 is used for UART/USART baud rate PCK5 is used for CAN PCK6 is used for TC
Real Time Clock					
RTCOUT0	Programmable RTC Waveform Output	Output	–	VDDIO	–
RTCOUT1	Programmable RTC Waveform Output	Output	–		–

Downloaded from Arrow.com.

Table 4-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Voltage Reference	Comments
Serial Wire Debug/JTAG Boundary Scan					
SWCLK/TCK	Serial Wire Clock / Test Clock (Boundary scan mode only)	Input	–	VDDIO	–
TDI	Test Data In (Boundary scan mode only)	Input	–		–
TDO/TRACESWO	Test Data Out (Boundary scan mode only)	Output	–		–
SWDIO/TMS	Serial Wire Input/Output / Test Mode Select (Boundary scan mode only)	I/O / Input	–		–
JTAGSEL	JTAG Selection	Input	High		–
Trace Debug Port					
TRACECLK	Trace Clock	Output	–	VDDIO	TRACECLK is PCK3
TRACED0–TRACED3	Trace Data	Output	–		
Flash Memory					
ERASE	Flash and NVM Configuration Bits Erase Command	Input	High	VDDIO	–
Reset/Test					
NRST	Synchronous Microcontroller Reset	I/O	Low	VDDIO	–
TST	Test Select	Input	–		–
Universal Asynchronous Receiver Transceiver - UART(x=[0:4])					
URXDx	UART Receive Data	Input	–	–	USPCK = PCK4 can be used to generate the baud rate
UTXDx	UART Transmit Data	Output	–	–	
PIO Controller - PIOA - PIOB - PIOC - PIOD - PIOE					
PA0–PA31	Parallel IO Controller A	I/O	–	VDDIO	–
PB0–PB9, PB12–PB13	Parallel IO Controller B	I/O	–		–
PC0– PC31	Parallel IO Controller C	I/O	–		–
PD0–PD31	Parallel IO Controller D	I/O	–	–	–
PE0–PE5	Parallel IO Controller E	I/O	–	–	–
PIO Controller - Parallel Capture Mode					
PIODC0–PIODC7	Parallel Capture Mode Data	Input	–	VDDIO	–
PIODCCLK	Parallel Capture Mode Clock	Input	–		–
PIODCEN1–PIODCEN2	Parallel Capture Mode Enable	Input	–		–
External Bus Interface					
D[15:0]	Data Bus	I/O	–	–	–
A[23:0]	Address Bus	Output	–	–	–
NWAIT	External Wait Signal	Input	Low	–	–

Downloaded from Arrow.com.

Table 4-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Voltage Reference	Comments
Static Memory Controller - SMC					
NCS0–NCS3	Chip Select Lines	Output	Low	–	–
NRD	Read Signal	Output	Low	–	–
NWE	Write Enable	Output	Low	–	–
NWR0–NWR1	Write Signal	Output	Low	–	–
NBS0–NBS1	Byte Mask Signal	Output	Low	–	Used also for SDRAMC
NAND Flash Logic					
NANDOE	NAND Flash Output Enable	Output	Low	–	–
NANDWE	NAND Flash Write Enable	Output	Low	–	–
SDR-SDRAM Controller Logic					
SDCK	SDRAM Clock	Output	–	–	–
SDCKE	SDRAM Clock Enable	Output	–	–	–
SDCS	SDRAM Controller Chip Select	Output	–	–	–
BA0–BA1	Bank Select	Output	–	–	–
SDWE	SDRAM Write Enable	Output	–	–	–
RAS–CAS	Row and Column Signal	Output	–	–	–
SDA10	SDRAM Address 10 Line	Output	–	–	–
High Speed Multimedia Card Interface - HSMCI					
MCCK	Multimedia Card Clock	I/O	–	–	–
MCCDA	Multimedia Card Slot A Command	I/O	–	–	–
MCDA0–MCDA3	Multimedia Card Slot A Data	I/O	–	–	–
Universal Synchronous Asynchronous Receiver Transmitter USART(x=[0:2])					
SCKx	USARTx Serial Clock	I/O	–	–	USPCK = PCK4 can be used to generate the baud rate
TXDx	USARTx Transmit Data	I/O	–	–	
RXDx	USARTx Receive Data	Input	–	–	
RTSx	USARTx Request To Send	Output	–	–	
CTSx	USARTx Clear To Send	Input	–	–	
DTRx	USARTx Data Terminal Ready	Output	–	–	
DSRx	USARTx Data Set Ready	Input	–	–	
DCDx	USARTx Data Carrier Detect	Input	–	–	
RIx	USARTx Ring Indicator	Input	–	–	
LONCOL1	LON Collision Detection	Input	–	–	
Synchronous Serial Controller - SSC					
TD	SSC Transmit Data	Output	–	–	–
RD	SSC Receive Data	Input	–	–	–
TK	SSC Transmit Clock	I/O	–	–	–

Downloaded from Arrow.com.

Table 4-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Voltage Reference	Comments
RK	SSC Receive Clock	I/O	–	–	–
TF	SSC Transmit Frame Sync	I/O	–	–	–
RF	SSC Receive Frame Sync	I/O	–	–	–
Inter-IC Sound Controller - I2SC[1..0]					
I2SCx_MCK	Master Clock	Output	–	VDDIO	–
I2SCx_CK	Serial Clock	I/O	–	VDDIO	–
I2SCx_WS	I2S Word Select	I/O	–	VDDIO	–
I2SCx_DI	Serial Data Input	Input	–	VDDIO	–
I2SCx_DO	Serial Data Output	Output	–	VDDIO	–
Image Sensor Interface - ISI					
ISI_D0–ISI_D11	Image Sensor Data	Input	–	–	–
ISI_MCK	Image sensor Reference clock. No dedicated signal, PCK1 can be used.	Output	–	–	–
ISI_HSYNC	Image Sensor Horizontal Synchro	Input	–	–	–
ISI_VSYNC	Image Sensor Vertical Synchro	Input	–	–	–
ISI_PCK	Image Sensor Data clock	Input	–	–	–
Timer/Counter - TC(x=[0:11])					
TCLKx	TC Channel x External Clock Input	Input	–	–	TCPCK = PCK6 can be used as an input clock
TIOAx	TC Channel x I/O Line A	I/O	–	–	
TIOBx	TC Channel x I/O Line B	I/O	–	–	
Pulse Width Modulation Controller- PWMC(x=[0..1])					
PWMCx_PWMH0–PWMCx_PWMH3	Waveform Output High for Channel 0–3	Output	–	–	–
PWMCx_PWML0–PWMCx_PWML3	Waveform Output Low for Channel 0–3	Output	–	–	Only output in complementary mode when dead time insertion is enabled.
PWMCx_PWMFI0–PWMCx_PWMFI2	Fault Input	Input	–	–	–
PWMCx_PWMEXTRG0–PWMCx_PWMEXTRG1	External Trigger Input	Input	–	–	–
Serial Peripheral Interface - SPI(x=[0..1])					
SPIx_MISO	Master In Slave Out	I/O	–	–	–
SPIx_MOSI	Master Out Slave In	I/O	–	–	–
SPIx_SPCK	SPI Serial Clock	I/O	–	–	–
SPIx_NPCS0	SPI Peripheral Chip Select 0	I/O	Low	–	–

Downloaded from Arrow.com.

Table 4-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Voltage Reference	Comments
SPIx_NPCS1–SPIx_NPCS3	SPI Peripheral Chip Select	Output	Low	–	–
Quad IO SPI - QSPI					
QSCK	QSPI Serial Clock	Output	–	–	–
QCS	QSPI Chip Select	Output	–	–	–
QIO0–QIO3	QSPI I/O QIO0 is QMOSI Master Out Slave In QIO1 is QMISO Master In Slave Out	I/O	–	–	–
Two-Wire Interface - TWIHS(x=0..2)					
TWDx	TWIx Two-wire Serial Data	I/O	–	–	–
TWCKx	TWIx Two-wire Serial Clock	I/O	–	–	–
Analog					
VREFP	ADC, DAC and Analog Comparator Positive Reference	Analog	–	–	–
VREFN	ADC, DAC and Analog Comparator Negative Reference Must be connected to GND or GNDANA.	Analog	–	–	–
12-bit Analog Front End - (x=[0..1])					
AFEx_AD0–AFEx_AD11	Analog Inputs	Analog, Digital	–	–	–
AFEx_ADTRG	ADC Trigger	Input	–	VDDIO	–
12-bit Digital-to-Analog Converter - DAC					
DAC0–DAC1	Analog Output	Analog, Digital	–	–	–
DATRG	DAC Trigger	Input	–	VDDIO	–
Fast Flash Programming Interface - FFPI					
PGMEN0–PGMEN1	Programming Enabling	Input	–	VDDIO	–
PGMM0–PGMM3	Programming Mode	Input	–	VDDIO	–
PGMD0–PGMD15	Programming Data	I/O	–		–
PGMRDY	Programming Ready	Output	High		–
PGMNVALID	Data Direction	Output	Low		–
PGMNOE	Programming Read	Input	Low		–
PGMNCMD	Programming Command	Input	Low		–
USB High Speed - USBHS					
HSDM	USB High Speed Data -	Analog, Digital	–	VDDUTMII	–
HSDP	USB High Speed Data +		–		–
VBG	Bias Voltage Reference for USB	Analog	–	–	–

Downloaded from Arrow.com.

Table 4-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Voltage Reference	Comments
Ethernet MAC 10/100 - GMAC					
GREFCK	Reference Clock	Input	–	–	RMII only
GTXCK	Transmit Clock	Input	–	–	MII only
GRXCK	Receive Clock	Input	–	–	MII only
GTXEN	Transmit Enable	Output	–	–	–
GTX0 - GTX3	Transmit Data	Output	–	–	GTX0–GTX1 only in RMII
GTXER	Transmit Coding Error	Output	–	–	MII only
GRXDV	Receive Data Valid	Input	–	–	MII only
GRX0 - GRX3	Receive Data	Input	–	–	GRX0–GRX1 only in RMII
GRXER	Receive Error	Input	–	–	–
GCRS	Carrier Sense	Input	–	–	MII only
GCOL	Collision Detected	Input	–	–	MII only
GMDC	Management Data Clock	Output	–	–	–
GMDIO	Management Data Input/Output	I/O	–	–	–
GTSUCOMP	TSU timer comparison valid	Output	–	–	–
Controller Area Network - CAN (x=[0:1])					
CANRXx	CAN Receive	Input	–	–	CANRX1 is available on PD28 for 100-pin only CANRX1 is available on PC12 for 144-pin only
CANTXx	CAN Transmit	Output	–	–	CANPCK = PCK5 can be used

Note: 1. Refer to Section 6.5 “Active Mode” for restrictions on the voltage range of analog cells.

Downloaded from Arrow.com.

5. Package and Pinout

In the tables that follow, the column “Reset State” indicates the reset state of the line with mnemonics.

- “PIO” “/” signal

Indicates whether the PIO Line resets in I/O mode or in peripheral mode. If “PIO” is mentioned, the PIO line is maintained in a static state as soon as the reset is released. As a result, the bit corresponding to the PIO line in the register PIO_PSR (Peripheral Status Register) resets low.

If a signal name is mentioned in the “Reset State” column, the PIO line is assigned to this function and the corresponding bit in PIO_PSR resets high. This is the case of pins controlling memories, in particular the address lines, which require the pin to be driven as soon as the reset is released.

- “I” / ”O”

Indicates whether the signal is input or output state.

- “PU” / “PD”

Indicates whether Pull-Up, Pull-Down or nothing is enabled.

- “ST”

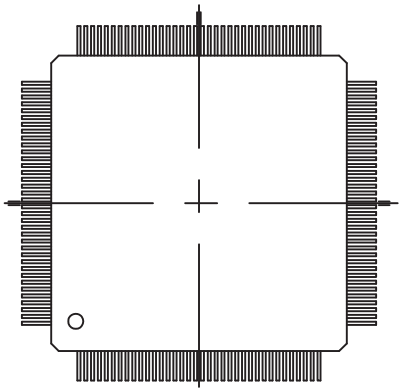
Indicates if Schmitt Trigger is enabled.

Downloaded from Arrow.com.

5.1 144-lead Packages

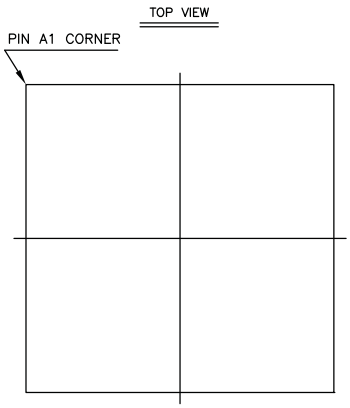
5.1.1 144-pin LQFP Package Outline

Figure 5-1. Orientation of the 144-pin LQFP Package



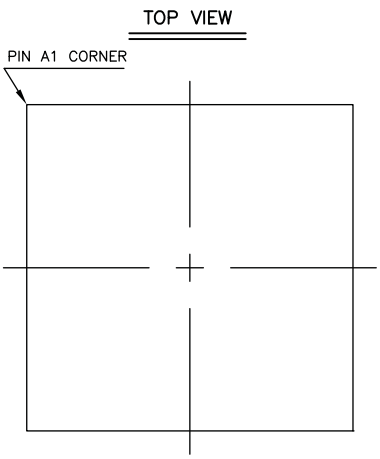
5.1.2 144-ball LFBGA Package Outline

Figure 5-2. Orientation of the 144-ball LFBGA Package



5.1.3 144-ball UFBGA Package Outline

Figure 5-3. Orientation of the 144-ball UFBGA Package



Downloaded from Arrow.com.

Downloaded from Arrow.com.

5.2 144-lead Package Pinout

Table 5-1. 144-lead Package Pinout

LQFP Pin	LFBGA Ball	UFBGA Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		PIO Peripheral D		Reset State
					Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, HiZ, ST
102	C11	E11	VDDIO	GPIO_AD	PA0	I/O	WKUP0[1]	I	PWMC0_PWMH0	O	TIOA0	I/O	A17/BA1	O	I2SC0_MCK	–	PIO, I, PU, ST
99	D12	F11	VDDIO	GPIO_AD	PA1	I/O	WKUP1[1]	I	PWMC0_PWML0	O	TIOB0	I/O	A18	O	I2SC0_CK	–	PIO, I, PU, ST
93	E12	G12	VDDIO	GPIO	PA2	I/O	WKUP2[1]	I	PWMC0_PWMH1	O	–	–	DATRG	I	–	–	PIO, I, PU, ST
91	F12	G11	VDDIO	GPIO_AD	PA3	I/O	PIODC0[2]	I	TWD0	I/O	LONCOL1	I	PCK2	O	–	–	PIO, I, PU, ST
77	K12	L12	VDDIO	GPIO	PA4	I/O	WKUP3/PIODC1[3]	I	TWCK0	O	TCLK0	I	UTXD1	O	–	–	PIO, I, PU, ST
73	M11	N13	VDDIO	GPIO_AD	PA5	I/O	WKUP4/PIODC2[3]	I	PWMC1_PWML3	O	ISI_D4	I	URXD1	I	–	–	PIO, I, PU, ST
114	B9	B11	VDDIO	GPIO_AD	PA6	I/O	–	–	–	–	PCK0	O	UTXD1	O	–	–	PIO, I, PU, ST
35	L2	N1	VDDIO	CLOCK	PA7	I/O	XIN32[4]	I	–	–	PWMC0_PWMH3	O	–	–	–	–	PIO, HiZ
36	M2	N2	VDDIO	CLOCK	PA8	I/O	XOUT32[4]	O	PWMC1_PWMH3	O	AFE0_ADTRG	I	–	–	–	–	PIO, HiZ
75	M12	L11	VDDIO	GPIO_AD	PA9	I/O	WKUP6/PIODC3[3]	I	URXD0	I	ISI_D3	I	PWMC0_PWMFI0	I	–	–	PIO, I, PU, ST
66	L9	M10	VDDIO	GPIO_AD	PA10	I/O	PIODC4[2]	I	UTXD0	O	PWMC0_PWMEXTRG0	I	RD	I	–	–	PIO, I, PU, ST
64	J9	N10	VDDIO	GPIO_AD	PA11	I/O	WKUP7/PIODC5[3]	I	QCS	O	PWMC0_PWMH0	O	PWMC1_PWML0	O	–	–	PIO, I, PU, ST
68	L10	N11	VDDIO	GPIO_AD	PA12	I/O	PIODC6[2]	I	QIO1	I/O	PWMC0_PWMH1	O	PWMC1_PWMH0	O	–	–	PIO, I, PU, ST
42	M3	M4	VDDIO	GPIO_AD	PA13	I/O	PIODC7[2]	I	QIO0	I/O	PWMC0_PWMH2	O	PWMC1_PWML1	O	–	–	PIO, I, PU, ST
51	K6	M6	VDDIO	GPIO_CLK	PA14	I/O	WKUP8/PIODCEN1[3]	I	QSCK	O	PWMC0_PWMH3	O	PWMC1_PWMH1	O	–	–	PIO, I, PU, ST
49	L5	N6	VDDIO	GPIO_AD	PA15	I/O	–	–	D14	I/O	TIOA1	I/O	PWMC0_PWML3	O	I2SC0_WS	–	PIO, I, PU, ST
45	K5	L4	VDDIO	GPIO_AD	PA16	I/O	–	–	D15	I/O	TIOB1	I/O	PWMC0_PWML2	O	I2SC0_DI	–	PIO, I, PU, ST
25	J1	J4	VDDIO	GPIO_AD	PA17	I/O	AFE0_AD6[5]	I	QIO2	I/O	PCK1	O	PWMC0_PWMH3	O	–	–	PIO, I, PU, ST
24	H2	J3	VDDIO	GPIO_AD	PA18	I/O	AFE0_AD7[5]	I	PWMC1_PWMEXTRG1	I	PCK2	O	A14	O	–	–	PIO, I, PU, ST
23	H1	J2	VDDIO	GPIO_AD	PA19	I/O	AFE0_AD8/WKUP9[6]	I	–	–	PWMC0_PWML0	O	A15	O	I2SC1_MCK	–	PIO, I, PU, ST
22	H3	J1	VDDIO	GPIO_AD	PA20	I/O	AFE0_AD9/WKUP10[6]	I	–	–	PWMC0_PWML1	O	A16/BA0	O	I2SC1_CK	–	PIO, I, PU, ST
32	K2	M1	VDDIO	GPIO_AD	PA21	I/O	AFE0_AD1/ PIODCEN2[8]	I	RXD1	I	PCK1	O	PWMC1_PWMFI0	I	–	–	PIO, I, PU, ST
37	K3	M2	VDDIO	GPIO_AD	PA22	I/O	PIODCCLK[2]	I	RK	I/O	PWMC0_PWMEXTRG1	I	NCS2	O	–	–	PIO, I, PU, ST
46	L4	N5	VDDIO	GPIO_AD	PA23	I/O	–	–	SCK1	I/O	PWMC0_PWMH0	O	A19	O	PWMC1_PWML2	O	PIO, I, PU, ST
56	L7	N8	VDDIO	GPIO_AD	PA24	I/O	–	–	RTS1	O	PWMC0_PWMH1	O	A20	O	ISI_PCK	I	PIO, I, PU, ST
59	K8	L8	VDDIO	GPIO_AD	PA25	I/O	–	–	CTS1	I	PWMC0_PWMH2	O	A23	O	MCCK	O	PIO, I, PU, ST
62	J8	M9	VDDIO	GPIO	PA26	I/O	–	–	DCD1	I	TIOA2	O	MCDA2	I/O	PWMC1_PWMFI1	I	PIO, I, PU, ST
70	J10	N12	VDDIO	GPIO_AD	PA27	I/O	–	–	DTR1	O	TIOB2	I/O	MCDA3	I/O	ISI_D7	I	PIO, I, PU, ST
112	C9	C11	VDDIO	GPIO	PA28	I/O	–	–	DSR1	I	TCLK1	I	MCCDA	I/O	PWMC1_PWMFI2	I	PIO, I, PU, ST
129	A6	A7	VDDIO	GPIO	PA29	I/O	–	–	RI1	I	TCLK2	I	–	–	–	–	PIO, I, PU, ST
116	A10	A11	VDDIO	GPIO	PA30	I/O	WKUP11[1]	I	PWMC0_PWML2	O	PWMC1_PWMEXTRG0	I	MCDA0	I/O	I2SC0_DO	–	PIO, I, PU, ST

Downloaded from Arrow.com.

Table 5-1. 144-lead Package Pinout (Continued)

LQFP Pin	LFBGA Ball	UFBGA Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		PIO Peripheral D		Reset State
					Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, HiZ, ST
118	C8	C10	VDDIO	GPIO_AD	PA31	I/O	–	–	SPI0_NPCS1	I/O	PCK2	O	MCDA1	I/O	PWMC1_PWMH2	O	PIO, I, PU, ST
21	H4	H2	VDDIO	GPIO	PB0	I/O	AFE0_AD10/ RTCOUT0[7]	I	PWMC0_PWMH0	O	–	–	RXD0	I	TF	I/O	PIO, I, PU, ST
20	G3	H1	VDDIO	GPIO	PB1	I/O	AFE1_AD0/ RTCOUT1[7]	I	PWMC0_PWMH1	O	GTSUCOMP	O	TXD0	I/O	TK	I/O	PIO, I, PU, ST
26	J2	K1	VDDIO	GPIO	PB2	I/O	AFE0_AD5[5]	I	CANTX0	O	–	–	CTS0	I	SPI0_NPCS0	I/O	PIO, I, PU, ST
31	J3	L1	VDDIO	GPIO_AD	PB3	I/O	AFE0_AD2/WKUP12[6]	I	CANRX0	I	PCK2	O	RTS0	O	ISI_D2	I	PIO, I, PU, ST
105	A12	C13	VDDIO	GPIO_MLB	PB4	I/O	TDI[9]	I	TWD1	I/O	PWMC0_PWMH2	O	–	–	TXD1	I/O	PIO, I, PD, ST
109	C10	C12	VDDIO	GPIO_MLB	PB5	I/O	TDO/TRACESWO/ WKUP13[9]	O	TWCK1	O	PWMC0_PWML0	O	–	–	TD	O	O, PU
79	J11	K11	VDDIO	GPIO	PB6	I/O	SWDIO/TMS[9]	I	–	–	–	–	–	–	–	–	PIO,I,ST
89	F9	H13	VDDIO	GPIO	PB7	I/O	SWCLK/TCK[9]	I	–	–	–	–	–	–	–	–	PIO,I,ST
141	A3	B2	VDDIO	CLOCK	PB8	I/O	XOUT[10]	O	–	–	–	–	–	–	–	–	PIO, HiZ
142	A2	A2	VDDIO	CLOCK	PB9	I/O	XIN[10]	I	–	–	–	–	–	–	–	–	PIO, HiZ
87	G12	J10	VDDIO	GPIO	PB12	I/O	ERASE[9]	I	PWMC0_PWML1	O	GTSUCOMP	O	–	–	PCK0	O	PIO, I, PD, ST
144	B2	A1	VDDIO	GPIO_AD	PB13	I/O	DAC0[11]	O	PWMC0_PWML2	O	PCK0	O	SCK0	I/O	–	–	PIO, I, PU, ST
11	E4	F2	VDDIO	GPIO_AD	PC0	I/O	AFE1_AD9[5]	I	D0	I/O	PWMC0_PWML0	O	–	–	–	–	PIO, I, PU, ST
38	J4	M3	VDDIO	GPIO_AD	PC1	I/O	–	–	D1	I/O	PWMC0_PWML1	O	–	–	–	–	PIO, I, PU, ST
39	K4	N3	VDDIO	GPIO_AD	PC2	I/O	–	–	D2	I/O	PWMC0_PWML2	O	–	–	–	–	PIO, I, PU, ST
40	L3	N4	VDDIO	GPIO_AD	PC3	I/O	–	–	D3	I/O	PWMC0_PWML3	O	–	–	–	–	PIO, I, PU, ST
41	J5	L3	VDDIO	GPIO_AD	PC4	I/O	–	–	D4	I/O	–	–	–	–	–	–	PIO, I, PU, ST
58	L8	M8	VDDIO	GPIO_AD	PC5	I/O	–	–	D5	I/O	TIOA6	I/O	–	–	–	–	PIO, I, PU, ST
54	K7	L7	VDDIO	GPIO_AD	PC6	I/O	–	–	D6	I/O	TIOB6	I/O	–	–	–	–	PIO, I, PU, ST
48	M4	L5	VDDIO	GPIO_AD	PC7	I/O	–	–	D7	I/O	TCLK6	I	–	–	–	–	PIO, I, PU, ST
82	J12	K13	VDDIO	GPIO_AD	PC8	I/O	–	–	NWR0/NWE	O	TIOA7	I/O	–	–	–	–	PIO, I, PU, ST
86	G11	J11	VDDIO	GPIO_AD	PC9	I/O	–	–	NANDOE	O	TIOB7	I/O	–	–	–	–	PIO, I, PU, ST
90	F10	H12	VDDIO	GPIO_AD	PC10	I/O	–	–	NANDWE	O	TCLK7	I	–	–	–	–	PIO, I, PU, ST
94	F11	F13	VDDIO	GPIO_AD	PC11	I/O	–	–	NRD	O	TIOA8	I/O	–	–	–	–	PIO, I, PU, ST
17	F4	G2	VDDIO	GPIO_AD	PC12	I/O	AFE1_AD3[5]	I	NCS3	O	TIOB8	I/O	CANRX1	I	–	–	PIO, I, PU, ST
19	G2	H3	VDDIO	GPIO_AD	PC13	I/O	AFE1_AD1[5]	I	NWAIT	I	PWMC0_PWMH3	O	SDA10	O	–	–	PIO, I, PU, ST
97	E10	F12	VDDIO	GPIO_AD	PC14	I/O	–	–	NCS0	O	TCLK8	I	CANTX1	O	–	–	PIO, I, PU, ST
18	G1	H4	VDDIO	GPIO_AD	PC15	I/O	AFE1_AD2[5]	I	NCS1/SDCS	O	PWMC0_PWML3	O	–	–	–	–	PIO, I, PU, ST
100	D11	E12	VDDIO	GPIO_AD	PC16	I/O	–	–	A21/NANDALE	O	–	–	–	–	–	–	PIO, I, PU, ST
103	B12	E10	VDDIO	GPIO_AD	PC17	I/O	–	–	A22/NANDCLE	O	–	–	–	–	–	–	PIO, I, PU, ST
111	B10	B12	VDDIO	GPIO_AD	PC18	I/O	–	–	A0/NBS0	O	PWMC0_PWML1	O	–	–	–	–	PIO, I, PU, ST

Downloaded from Arrow.com.

Table 5-1. 144-lead Package Pinout (Continued)

LQFP Pin	LFBGA Ball	UFBGA Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		PIO Peripheral D		Reset State
					Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, HiZ, ST
117	D8	B10	VDDIO	GPIO_AD	PC19	I/O	–	–	A1	O	PWMC0_PWMH2	O	–	–	–	–	PIO, I, PU, ST
120	A9	C9	VDDIO	GPIO_AD	PC20	I/O	–	–	A2	O	PWMC0_PWML2	O	–	–	–	–	PIO, I, PU, ST
122	A7	A9	VDDIO	GPIO_AD	PC21	I/O	–	–	A3	O	PWMC0_PWMH3	O	–	–	–	–	PIO, I, PU, ST
124	C7	A8	VDDIO	GPIO_AD	PC22	I/O	–	–	A4	O	PWMC0_PWML3	O	–	–	–	–	PIO, I, PU, ST
127	C6	C7	VDDIO	GPIO_AD	PC23	I/O	–	–	A5	O	TIOA3	I/O	–	–	–	–	PIO, I, PU, ST
130	B6	D7	VDDIO	GPIO_AD	PC24	I/O	–	–	A6	O	TIOB3	I/O	SPI1_SPCK	O	–	–	PIO, I, PU, ST
133	C5	C6	VDDIO	GPIO_AD	PC25	I/O	–	–	A7	O	TCLK3	I	SPI1_NPCS0	I/O	–	–	PIO, I, PU, ST
13	F2	F4	VDDIO	GPIO_AD	PC26	I/O	AFE1_AD7[5]	I	A8	O	TIOA4	I/O	SPI1_MISO	I	–	–	PIO, I, PU, ST
12	E2	F3	VDDIO	GPIO_AD	PC27	I/O	AFE1_AD8[5]	I	A9	O	TIOB4	I/O	SPI1_MOSI	O	–	–	PIO, I, PU, ST
76	L12	L13	VDDIO	GPIO_AD	PC28	I/O	–	–	A10	O	TCLK4	I	SPI1_NPCS1	I/O	–	–	PIO, I, PU, ST
16	F3	G1	VDDIO	GPIO_AD	PC29	I/O	AFE1_AD4[5]	I	A11	O	TIOA5	I/O	SPI1_NPCS2	O	–	–	PIO, I, PU, ST
15	F1	G3	VDDIO	GPIO_AD	PC30	I/O	AFE1_AD5[5]	I	A12	O	TIOB5	I/O	SPI1_NPCS3	O	–	–	PIO, I, PU, ST
14	E1	G4	VDDIO	GPIO_AD	PC31	I/O	AFE1_AD6[5]	I	A13	O	TCLK5	I	–	–	–	–	PIO, I, PU, ST
1	D4	B1	VDDIO	GPIO_AD	PD0	I/O	DAC1[11]	I	GTXCK	I	PWMC1_PWML0	O	SPI1_NPCS1	I/O	DCD0	I	PIO, I, PU, ST
132	B5	B6	VDDIO	GPIO	PD1	I/O	–	–	GTXEN	O	PWMC1_PWMH0	O	SPI1_NPCS2	I/O	DTR0	O	PIO, I, PU, ST
131	A5	A6	VDDIO	GPIO	PD2	I/O	–	–	GTX0	O	PWMC1_PWML1	O	SPI1_NPCS3	I/O	DSR0	I	PIO, I, PU, ST
128	B7	B7	VDDIO	GPIO	PD3	I/O	–	–	GTX1	O	PWMC1_PWMH1	O	UTXD4	O	RI0	I	PIO, I, PU, ST
126	D6	C8	VDDIO	GPIO_CLK	PD4	I/O	–	–	GRXDV	I	PWMC1_PWML2	O	TRACED0	O	DCD2	I	PIO, I, PU, ST
125	D7	B8	VDDIO	GPIO_CLK	PD5	I/O	–	–	GRX0	I	PWMC1_PWMH2	O	TRACED1	O	DTR2	O	PIO, I, PU, ST
121	A8	B9	VDDIO	GPIO_CLK	PD6	I/O	–	–	GRX1	I	PWMC1_PWML3	O	TRACED2	O	DSR2	I	PIO, I, PU, ST
119	B8	A10	VDDIO	GPIO_CLK	PD7	I/O	–	–	GRXER	I	PWMC1_PWMH3	O	TRACED3	O	RI2	I	PIO, I, PU, ST
113	E9	A12	VDDIO	GPIO_CLK	PD8	I/O	–	–	GMDC	O	PWMC0_PWMFI1	I	–	–	TRACECLK	O	PIO, I, PU, ST
110	D9	A13	VDDIO	GPIO_CLK	PD9	I/O	–	–	GMDIO	I/O	PWMC0_PWMFI2	I	AFE1_ADTRG	I	–	–	PIO, I, PU, ST
101	C12	D13	VDDIO	GPIO_MLB	PD10	I/O	–	–	GCRS	I	PWMC0_PWML0	O	TD	O	–	–	PIO, I, PD, ST
98	E11	E13	VDDIO	GPIO_AD	PD11	I/O	–	–	GRX2	I	PWMC0_PWMH0	O	GTSUCOMP	O	ISI_D5	I	PIO, I, PU, ST
92	G10	G13	VDDIO	GPIO_AD	PD12	I/O	–	–	GRX3	I	CANTX1	O	SPI0_NPCS2	O	ISI_D6	I	PIO, I, PU, ST
88	G9	H11	VDDIO	GPIO_CLK	PD13	I/O	–	–	GCOL	I	–	–	SDA10	O	–	–	PIO, I, PU, ST
84	H10	J12	VDDIO	GPIO_AD	PD14	I/O	–	–	GRXCK	I	–	–	SDCKE	O	–	–	PIO, I, PU, ST
106	A11	D11	VDDIO	GPIO_AD	PD15	I/O	–	–	GTX2	O	RXD2	I	NWR1/NBS1	O	–	–	PIO, I, PU, ST
78	K11	K10	VDDIO	GPIO_AD	PD16	I/O	–	–	GTX3	O	TXD2	I/O	RAS	O	–	–	PIO, I, PU, ST
74	L11	M13	VDDIO	GPIO_AD	PD17	I/O	–	–	GTXER	O	SCK2	I/O	CAS	O	–	–	PIO, I, PU, ST
69	M10	M11	VDDIO	GPIO_AD	PD18	I/O	–	–	NCS1/SDCS	O	RTS2	O	URXD4	I	–	–	PIO, I, PU, ST
67	M9	L10	VDDIO	GPIO_AD	PD19	I/O	–	–	NCS3	O	CTS2	I	UTXD4	O	–	–	PIO, I, PU, ST

Downloaded from Arrow.com.

Table 5-1. 144-lead Package Pinout (Continued)

LQFP Pin	LFBGA Ball	UFBGA Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		PIO Peripheral D		Reset State
					Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, HiZ, ST
65	K9	K9	VDDIO	GPIO	PD20	I/O	–	–	PWMC0_PWMH0	O	SPI0_MISO	I/O	GTSUCOMP	O	–	–	PIO, I, PU, ST
63	H9	L9	VDDIO	GPIO_AD	PD21	I/O	–	–	PWMC0_PWMH1	O	SPI0_MOSI	I/O	TIOA11	I/O	ISI_D1	I	PIO, I, PU, ST
60	M8	N9	VDDIO	GPIO_AD	PD22	I/O	–	–	PWMC0_PWMH2	O	SPI0_SPCK	O	TIOB11	I/O	ISI_D0	I	PIO, I, PU, ST
57	M7	N7	VDDIO	GPIO_CLK	PD23	I/O	–	–	PWMC0_PWMH3	O	–	–	SDCK	O	–	–	PIO, I, PU, ST
55	M6	K7	VDDIO	GPIO_AD	PD24	I/O	–	–	PWMC0_PWML0	O	RF	I/O	TCLK11	I	ISI_HSYNC	I	PIO, I, PU, ST
52	M5	L6	VDDIO	GPIO_AD	PD25	I/O	–	–	PWMC0_PWML1	O	SPI0_NPCS1	I/O	URXD2	I	ISI_VSYNC	I	PIO, I, PU, ST
53	L6	M7	VDDIO	GPIO	PD26	I/O	–	–	PWMC0_PWML2	O	TD	O	UTXD2	O	UTXD1	O	PIO, I, PU, ST
47	J6	M5	VDDIO	GPIO_AD	PD27	I/O	–	–	PWMC0_PWML3	O	SPI0_NPCS3	O	TWD2	O	ISI_D8	I	PIO, I, PU, ST
71	K10	M12	VDDIO	GPIO_AD	PD28	I/O	WKUP5[1]	I	URXD3	I	CANRX1	I	TWCK2	O	ISI_D9	I	PIO, I, PU, ST
108	D10	B13	VDDIO	GPIO_AD	PD29	I/O	–	–	–	–	–	–	SDWE	O	–	–	PIO, I, PU, ST
34	M1	L2	VDDIO	GPIO_AD	PD30	I/O	AFE0_AD0[5]	I	UTXD3	0	–	–	–	–	ISI_D10	I	PIO, I, PU, ST
2	D3	C3	VDDIO	GPIO_AD	PD31	I/O	–	–	QIO3	I/O	UTXD3	O	PCK2	O	ISI_D11	I	PIO, I, PU, ST
4	C2	C2	VDDIO	GPIO_AD	PE0	I/O	AFE1_AD11[5]	I	D8	I/O	TIOA9	I/O	I2SC1_WS	–	–	–	PIO, I, PU, ST
6	A1	D2	VDDIO	GPIO_AD	PE1	I/O	–	–	D9	I/O	TIOB9	I/O	I2SC1_DO	–	–	–	PIO, I, PU, ST
7	B1	D1	VDDIO	GPIO_AD	PE2	I/O	–	–	D10	I/O	TCLK9	I	I2SC1_DI	–	–	–	PIO, I, PU, ST
10	E3	F1	VDDIO	GPIO_AD	PE3	I/O	AFE1_AD10[5]	I	D11	I/O	TIOA10	I/O	–	–	–	–	PIO, I, PU, ST
27	K1	K2	VDDIO	GPIO_AD	PE4	I/O	AFE0_AD4[5]	I	D12	I/O	TIOB10	I/O	–	–	–	–	PIO, I, PU, ST
28	L1	K3	VDDIO	GPIO_AD	PE5	I/O	AFE0_AD3[5]	I	D13	I/O	TCLK10	I/O	–	–	–	–	PIO, I, PU, ST
3	C3	E4	VDDOUT	Power	VDDOUT	–	–	–	–	–	–	–	–	–	–	–	–
5	C1	C1	VDDIN	Power	VDDIN	–	–	–	–	–	–	–	–	–	–	–	–
8	D2	E2	GND	Reference	VREFN	I	–	–	–	–	–	–	–	–	–	–	–
9	D1	E1	VDDIO	Reference	VREFP	I	–	–	–	–	–	–	–	–	–	–	–
83	H12	K12	VDDIO	RST	NRST	I/O	–	–	–	–	–	–	–	–	–	–	I, PU
85	H11	J13	VDDIO	TEST	TST	I	–	–	–	–	–	–	–	–	–	–	I, PD
30,43,72, 80,96	G8, H6, H7	D6, F10, K6	VDDIO	Power	VDDIO	–	–	–	–	–	–	–	–	–	–	–	–
104	B11	D12	VDDIO	TEST	JTAGSEL	I	–	–	–	–	–	–	–	–	–	–	I, PD
29,33,50, 81,107	E8, H5, H8	D5, G10, K5	VDDCORE	Power	VDDCORE	–	–	–	–	–	–	–	–	–	–	–	–
123	J7	D8	VDDPLL	Power	VDDPLL	–	–	–	–	–	–	–	–	–	–	–	–
134	E7	B4	VDDUTMII	Power	VDDUTMII	–	–	–	–	–	–	–	–	–	–	–	–
136	B4	A5	VDDUTMII	USBHS	HSDM	I/O	–	–	–	–	–	–	–	–	–	–	–
137	A4	A4	VDDUTMII	USBHS	HSDP	I/O	–	–	–	–	–	–	–	–	–	–	–

Downloaded from Arrow.com.

Table 5-1. 144-lead Package Pinout (Continued)

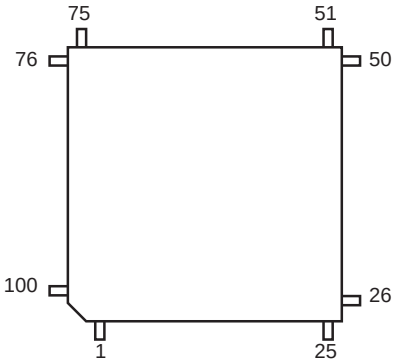
LQFP Pin	LFBGA Ball	UFBGA Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		PIO Peripheral D		Reset State
					Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, HiZ, ST
44,61,95, 115,135,138	F5, F6, G4, G5, G6, G7	C5, D3, D10, H10, K4, K8	GND	Ground	GND	–	–	–	–	–	–	–	–	–	–	–	–
–	D5	E3	GNDANA	Ground	GNDANA	–	–	–	–	–	–	–	–	–	–	–	–
–	E5	B5	GNDUTMI	Ground	GNDUTMI	–	–	–	–	–	–	–	–	–	–	–	–
–	E6	B3	GNDPLLUSB	Ground	GNDPLLUSB	–	–	–	–	–	–	–	–	–	–	–	–
–	F7	D9	GNDPLL	Ground	GNDPLL	–	–	–	–	–	–	–	–	–	–	–	–
139	B3	C4	VDDUTMIC	Power	VDDUTMIC	–	–	–	–	–	–	–	–	–	–	–	–
140	C4	A3	–	VBG	VBG	I	–	–	–	–	–	–	–	–	–	–	–
143	F8	D4	VDDPLLUSB	Power	VDDPLLUSB	–	–	–	–	–	–	–	–	–	–	–	–

Notes:
1. WKUPx can be used if the PIO Controller defines the I/O line as "input".
2. To select this extra function, refer to Section 32.5.14 “Parallel Capture Mode”.
3. PIODCEN1/PIODCx has priority over WKUPx. Refer to Section 32.5.14 “Parallel Capture Mode”.
4. Refer to Section 22.4.2 “Slow Clock Generator”.
5. To select this extra function, refer to Section 50.5.1 “I/O Lines”.
6. Analog input has priority over WKUPx pin. To select the analog input, refer to Section 50.5.1 “I/O Lines”. WKUPx can be used if the PIO controller defines the I/O line as "input".
7. Analog input has priority over RTCOUTx pin. To select the analog input, refer to Section 50.5.1 “I/O Lines”. Refer to Section 26.5.8 “Waveform Generation” to select RTCOUTx.
8. Analog input has priority over WKUPx pin. To select the analog input, refer to Section 50.5.1 “I/O Lines”. To select PIODCEN2, refer to Section 32.5.14 “Parallel Capture Mode”.
9. Refer to the System I/O Configuration Register in Section 18. “Bus Matrix (MATRIX)”.
10. Refer to Section 30.5.3 “3 to 20 MHz Crystal or Ceramic Resonator-based Oscillator”.
11. DAC0 is selected when DACC_CHER.CH0 is set. DAC1 is selected when DACC_CHER.CH1 is set. Refer to Section 51.7.4 “DACC Channel Enable Register”.

5.3 100-lead Packages

5.3.1 100-pin LQFP Package Outline

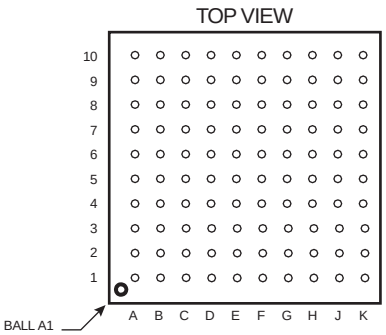
Figure 5-4. Orientation of the 100-lead LQFP Package



5.3.2 100-ball TFBGA Package Outline

The 100-ball TFBGA package has a 0.8 mm ball pitch and respects Green standards. Its dimensions are 9 x 9 x 1.1 mm. Figure 5-5 shows the orientation of the 100-ball TFBGA Package.

Figure 5-5. Orientation of the 100-ball TFBGA Package



Downloaded from Arrow.com.

Downloaded from Arrow.com.

5.4 100-lead Package Pinout

Table 5-2. 100-lead Package Pinout

LQFP Pin	TFBGA Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		PIO Peripheral D		Reset State
				Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, HiZ, ST
72	D8	VDDIO	GPIO_AD	PA0	I/O	WKUP0[1]	I	PWMC0_PWMH0	O	TIOA0	I/O	A17/BA1	O	I2SC0_MCK	–	PIO, I, PU, ST
70	C10	VDDIO	GPIO_AD	PA1	I/O	WKUP1[1]	I	PWMC0_PWML0	O	TIOB0	I/O	A18	O	I2SC0_CK	–	PIO, I, PU, ST
66	D10	VDDIO	GPIO	PA2	I/O	WKUP2[1]	I	PWMC0_PWMH1	O	–	–	DATRG	I	–	–	PIO, I, PU, ST
64	F9	VDDIO	GPIO_AD	PA3	I/O	PIODC0[2]	I	TWD0	I/O	LONCOL1	I	PCK2	O	–	–	PIO, I, PU, ST
55	H10	VDDIO	GPIO	PA4	I/O	WKUP3/PIODC1[3]	I	TWCK0	O	TCLK0	I	UTXD1	O	–	–	PIO, I, PU, ST
52	H9	VDDIO	GPIO_AD	PA5	I/O	WKUP4/PIODC2[3]	I	PWMC1_PWML3	O	ISI_D4	I	URXD1	I	–	–	PIO, I, PU, ST
24	J2	VDDIO	CLOCK	PA7	I/O	XIN32[4]	I	–	–	PWMC0_PWMH3	–	–	–	–	–	PIO, HiZ
25	K2	VDDIO	CLOCK	PA8	I/O	XOUT32[4]	O	PWMC1_PWMH3	O	AFE0_ADTRG	I	–	–	–	–	PIO, HiZ
54	J9	VDDIO	GPIO_AD	PA9	I/O	WKUP6/PIODC3[3]	I	URXD0	I	ISI_D3	I	PWMC0_PWMFI0	I	–	–	PIO, I, PU, ST
46	K9	VDDIO	GPIO_AD	PA10	I/O	PIODC4[2]	I	UTXD0	O	PWMC0_PWMEXTRG0	I	RD	I	–	–	PIO, I, PU, ST
44	J8	VDDIO	GPIO_AD	PA11	I/O	WKUP7/PIODC5[3]	I	QCS	O	PWMC0_PWMH0	O	PWMC1_PWML0	O	–	–	PIO, I, PU, ST
48	K10	VDDIO	GPIO_AD	PA12	I/O	PIODC6[2]	I	QIO1	I/O	PWMC0_PWMH1	O	PWMC1_PWMH0	O	–	–	PIO, I, PU, ST
27	G5	VDDIO	GPIO_AD	PA13	I/O	PIODC7[2]	I	QIO0	I/O	PWMC0_PWMH2	O	PWMC1_PWML1	O	–	–	PIO, I, PU, ST
34	H6	VDDIO	GPIO_CLK	PA14	I/O	WKUP8/PIODCEN1[3]	I	QSCK	O	PWMC0_PWMH3	O	PWMC1_PWMH1	O	–	–	PIO, I, PU, ST
33	J6	VDDIO	GPIO_AD	PA15	I/O	–	–	D14	I/O	TIOA1	I/O	PWMC0_PWML3	O	I2SC0_WS	–	PIO, I, PU, ST
30	J5	VDDIO	GPIO_AD	PA16	I/O	–	–	D15	I/O	TIOB1	I/O	PWMC0_PWML2	O	I2SC0_DI	–	PIO, I, PU, ST
16	G1	VDDIO	GPIO_AD	PA17	I/O	AFE0_AD6[5]	I	QIO2	I/O	PCK1	O	PWMC0_PWMH3	O	–	–	PIO, I, PU, ST
15	G2	VDDIO	GPIO_AD	PA18	I/O	AFE0_AD7[5]	I	PWMC1_PWMEXTRG1	I	PCK2	O	A14	O	–	–	PIO, I, PU, ST
14	F1	VDDIO	GPIO_AD	PA19	I/O	AFE0_AD8/WKUP9[6]	I	–	–	PWMC0_PWML0	O	A15	O	I2SC1_MCK	–	PIO, I, PU, ST
13	F2	VDDIO	GPIO_AD	PA20	I/O	AFE0_AD9/WKUP10[6]	I	–	–	PWMC0_PWML1	O	A16/BA0	O	I2SC1_CK	–	PIO, I, PU, ST
21	J1	VDDIO	GPIO_AD	PA21	I/O	AFE0_AD1/ PIODCEN2[8]	I	RXD1	I	PCK1	O	PWMC1_PWMFI0	I	–	–	PIO, I, PU, ST
26	J3	VDDIO	GPIO_AD	PA22	I/O	PIODCCLK[2]	I	RK	I/O	PWMC0_PWMEXTRG1	I	NCS2	O	–	–	PIO, I, PU, ST
31	K5	VDDIO	GPIO_AD	PA23	I/O	–	–	SCK1	I/O	PWMC0_PWMH0	O	A19	O	PWMC1_PWML2	O	PIO, I, PU, ST
38	K7	VDDIO	GPIO_AD	PA24	I/O	–	–	RTS1	O	PWMC0_PWMH1	O	A20	O	ISI_PCK	I	PIO, I, PU, ST
40	H7	VDDIO	GPIO_AD	PA25	I/O	–	–	CTS1	I	PWMC0_PWMH2	O	A23	O	MCCK	O	PIO, I, PU, ST
42	K8	VDDIO	GPIO	PA26	I/O	–	–	DCD1	I	TIOA2	O	MCDA2	I/O	PWMC1_PWMFI1	I	PIO, I, PU, ST
50	H8	VDDIO	GPIO_AD	PA27	I/O	–	–	DTR1	O	TIOB2	I/O	MCDA3	I/O	ISI_D7	I	PIO, I, PU, ST
79	A9	VDDIO	GPIO	PA28	I/O	–	–	DSR1	I	TCLK1	I	MCCDA	I/O	PWMC1_PWMFI2	I	PIO, I, PU, ST
82	C7	VDDIO	GPIO	PA30	I/O	WKUP11[1]	I	PWMC0_PWML2	O	PWMC1_PWMEXTRG0	I	MCDA0	I/O	I2SC0_DO	–	PIO, I, PU, ST
83	A7	VDDIO	GPIO_AD	PA31	I/O	–	–	SPI0_NPCS1	I/O	PCK2	O	MCDA1	I/O	PWMC1_PWMH2	O	PIO, I, PU, ST

Downloaded from Arrow.com.

Table 5-2. 100-lead Package Pinout

LQFP Pin	TFBGA Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		PIO Peripheral D		Reset State
				Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, HiZ, ST
12	E1	VDDIO	GPIO	PB0	I/O	AFE0_AD10/ RTCOUT0[7]	I	PWMC0_PWMH0	O	–	–	RXD0	I	TF	I/O	PIO, I, PU, ST
11	E2	VDDIO	GPIO	PB1	I/O	AFE1_AD0/ RTCOUT1[7]	I	PWMC0_PWMH1	O	–	O	TXD0	I/O	TK	I/O	PIO, I, PU, ST
17	H1	VDDIO	GPIO	PB2	I/O	AFE0_AD5[5]	I	CANTX0	O	–	–	CTS0	I	SPI0_NPCS0	I/O	PIO, I, PU, ST
20	H2	VDDIO	GPIO_AD	PB3	I/O	AFE0_AD2/WKUP12[6]	I	CANRX0	I	PCK2	O	RTS0	O	ISI_D2	I	PIO, I, PU, ST
74	B9	VDDIO	GPIO_MLB	PB4	I/O	TDI[9]	I	TWD1	I/O	PWMC0_PWMH2	O	–	–	TXD1	I/O	PIO, I, PD, ST
77	C8	VDDIO	GPIO_MLB	PB5	I/O	TDO/TRACESWO/ WKUP13[9]	O	TWCK1	O	PWMC0_PWML0	O	–	–	TD	O	O, PU
57	G8	VDDIO	GPIO	PB6	I/O	SWDIO/TMS[9]	I	–	–	–	–	–	–	–	–	PIO,I,ST
63	E9	VDDIO	GPIO	PB7	I/O	SWCLK/TCK[9]	I	–	–	–	–	–	–	–	–	PIO,I,ST
98	A2	VDDIO	CLOCK	PB8	I/O	XOUT[10]	O	–	–	–	–	–	–	–	–	PIO, HiZ
99	A1	VDDIO	CLOCK	PB9	I/O	XIN[10]	I	–	–	–	–	–	–	–	–	PIO, HiZ
61	F8	VDDIO	GPIO	PB12	I/O	ERASE[9]	I	PWMC0_PWML1	O	GTSUCOMP	O	–	–	PCK0	O	PIO, I, PD, ST
100	B2	VDDIO	GPIO_AD	PB13	I/O	DAC0[11]	O	PWMC0_PWML2	O	PCK0	O	SCK0	I/O	–	–	PIO, I, PU, ST
1	C1	VDDIO	GPIO_AD	PD0	I/O	DAC1[11]	I	GTXCK	I	PWMC1_PWML0	O	SPI1_NPCS1	I/O	DCD0	I	PIO, I, PU, ST
92	D2	VDDIO	GPIO	PD1	I/O	–	–	GTXEN	O	PWMC1_PWMH0	O	SPI1_NPCS2	I/O	DTR0	O	PIO, I, PU, ST
91	E3	VDDIO	GPIO	PD2	I/O	–	–	GTX0	O	PWMC1_PWML1	O	SPI1_NPCS3	I/O	DSR0	I	PIO, I, PU, ST
89	B5	VDDIO	GPIO	PD3	I/O	–	–	GTX1	O	PWMC1_PWMH1	O	UTXD4	O	RI0	I	PIO, I, PU, ST
88	A5	VDDIO	GPIO_CLK	PD4	I/O	–	–	GRXDV	I	PWMC1_PWML2	O	TRACED0	O	DCD2	I	PIO, I, PU, ST
87	D5	VDDIO	GPIO_CLK	PD5	I/O	–	–	GRX0	I	PWMC1_PWMH2	O	TRACED1	O	DTR2	O	PIO, I, PU, ST
85	B6	VDDIO	GPIO_CLK	PD6	I/O	–	–	GRX1	I	PWMC1_PWML3	O	TRACED2	O	DSR2	I	PIO, I, PU, ST
84	A6	VDDIO	GPIO_CLK	PD7	I/O	–	–	GRXER	I	PWMC1_PWMH3	O	TRACED3	O	RI2	I	PIO, I, PU, ST
80	B7	VDDIO	GPIO_CLK	PD8	I/O	–	–	GMDC	O	PWMC0_PWMFI1	I	–	–	TRACECLK	O	PIO, I, PU, ST
78	B8	VDDIO	GPIO_CLK	PD9	I/O	–	–	GMDIO	I/O	PWMC0_PWMFI2	I	AFE1_ADTRG	I	–	–	PIO, I, PU, ST
71	C9	VDDIO	GPIO_MLB	PD10	I/O	–	–	GCRS	I	PWMC0_PWML0	O	TD	O	–	–	PIO, I, PD, ST
69	D9	VDDIO	GPIO_AD	PD11	I/O	–	–	GRX2	I	PWMC0_PWMH0	O	GTSUCOMP	O	ISI_D5	I	PIO, I, PU, ST
65	E10	VDDIO	GPIO_AD	PD12	I/O	–	–	GRX3	I	CANTX1	O	SPI0_NPCS2	O	ISI_D6	I	PIO, I, PU, ST
62	E8	VDDIO	GPIO_AD	PD13	I/O	–	–	GCOL	I	–	–	SDA10	O	–	–	PIO, I, PU, ST
59	F10	VDDIO	GPIO_AD	PD14	I/O	–	–	GRXCK	I	–	–	SDCKE	O	–	–	PIO, I, PU, ST
75	B10	VDDIO	GPIO_AD	PD15	I/O	–	–	GTX2	O	RXD2	I	NWR1/NBS1	O	–	–	PIO, I, PU, ST
56	G9	VDDIO	GPIO_AD	PD16	I/O	–	–	GTX3	O	TXD2	I/O	RAS	O	–	–	PIO, I, PU, ST
53	J10	VDDIO	GPIO_AD	PD17	I/O	–	–	GTXER	O	SCK2	I/O	CAS	O	–	–	PIO, I, PU, ST
49	K6	VDDIO	GPIO_AD	PD18	I/O	–	–	NCS1/SDCS	O	RTS2	O	URXD4	I	–	–	PIO, I, PU, ST
47	K4	VDDIO	GPIO_AD	PD19	I/O	–	–	NCS3	O	CTS2	I	UTXD4	O	–	–	PIO, I, PU, ST

Downloaded from Arrow.com.

Table 5-2. 100-lead Package Pinout

LQFP Pin	TFBGA Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		PIO Peripheral D		Reset State
				Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, HiZ, ST
45	K3	VDDIO	GPIO	PD20	I/O	–	–	PWMC0_PWMH0	O	SPI0_MISO	I/O	GTSUCOMP	O	–	–	PIO, I, PU, ST
43	H5	VDDIO	GPIO_AD	PD21	I/O	–	–	PWMC0_PWMH1	O	SPI0_MOSI	I/O	TIOA11	I/O	ISI_D1	I	PIO, I, PU, ST
41	J4	VDDIO	GPIO_AD	PD22	I/O	–	–	PWMC0_PWMH2	O	SPI0_SPCK	O	TIOB11	I/O	ISI_D0	I	PIO, I, PU, ST
37	G4	VDDIO	GPIO_AD	PD24	I/O	–	–	PWMC0_PWML0	O	RF	I/O	TCLK11	I	ISI_HSYNC	I	PIO, I, PU, ST
35	H3	VDDIO	GPIO_AD	PD25	I/O	–	–	PWMC0_PWML1	O	SPI0_NPCS1	I/O	URXD2	I	ISI_VSYNC	I	PIO, I, PU, ST
36	G3	VDDIO	GPIO	PD26	I/O	–	–	PWMC0_PWML2	O	TD	O	UTXD2	O	UTXD1	O	PIO, I, PU, ST
32	H4	VDDIO	GPIO_AD	PD27	I/O	–	–	PWMC0_PWML3	O	SPI0_NPCS3	O	TWD2	O	ISI_D8	I	PIO, I, PU, ST
51	J7	VDDIO	GPIO_AD	PD28	I/O	WKUP5[1]	I	URXD3	I	CANRX1	I	TWCK2	O	ISI_D9	I	PIO, I, PU, ST
23	K1	VDDIO	GPIO_AD	PD30	I/O	AFE0_AD0[5]	I	UTXD3	0	–	–	–	–	ISI_D10	I	PIO, I, PU, ST
2	B1	VDDIO	GPIO_AD	PD31	I/O	–	–	QIO3	I/O	UTXD3	O	PCK2	O	ISI_D11	I	PIO, I, PU, ST
4	C3	VDDOUT	Power	VDDOUT	–	–	–	–	–	–	–	–	–	–	–	–
5	C2	VDDIN	Power	VDDIN	–	–	–	–	–	–	–	–	–	–	–	–
6	D3	GND	Reference	VREFN	I	–	–	–	–	–	–	–	–	–	–	–
9	D1	VDDIO	Reference	VREFP	I	–	–	–	–	–	–	–	–	–	–	–
58	G10	VDDIO	RST	NRST	I/O	–	–	–	–	–	–	–	–	–	–	I, PU
60	F7	VDDIO	TEST	TST	I	–	–	–	–	–	–	–	–	–	–	I, PD
19, 28, 68, 81	C5, F3, G7	VDDIO	Power	VDDIO	–	–	–	–	–	–	–	–	–	–	–	–
73	A10	VDDIO	TEST	JTAGSEL	I	–	–	–	–	–	–	–	–	–	–	I, PD
18, 22, 39, 76	C6, D6, G6	VDDCORE	Power	VDDCORE	–	–	–	–	–	–	–	–	–	–	–	–
86	D7	VDDPLL	Power	VDDPLL	–	–	–	–	–	–	–	–	–	–	–	–
93	E5	VDDUTMII	Power	VDDUTMII	–	–	–	–	–	–	–	–	–	–	–	–
94	A4	VDDUTMII	USBHS	HSDM	I/O	–	–	–	–	–	–	–	–	–	–	–
95	B4	VDDUTMII	USBHS	HSDP	I/O	–	–	–	–	–	–	–	–	–	–	–
3, 7, 8, 10, 29, 67	E7, F4, F5, F6	GND	Ground	GND	–	–	–	–	–	–	–	–	–	–	–	–
–	D4	GNDANA	Ground	GNDANA	–	–	–	–	–	–	–	–	–	–	–	–
–	A8	GNDUTMI	Ground	GNDUTMI	–	–	–	–	–	–	–	–	–	–	–	–
–	C4	GNDPLLUSB	Ground	GNDPLLUSB	–	–	–	–	–	–	–	–	–	–	–	–
–	E4	GNDPLL	Ground	GNDPLLUSB	–	–	–	–	–	–	–	–	–	–	–	–
96	B3	VDDUTMIC	Power	VDDUTMIC	–	–	–	–	–	–	–	–	–	–	–	–
97	A3	–	VBG	VBG	I	–	–	–	–	–	–	–	–	–	–	–
90	E6	VDDPLLUSB	Power	VDDPLLUSB	–	–	–	–	–	–	–	–	–	–	–	–

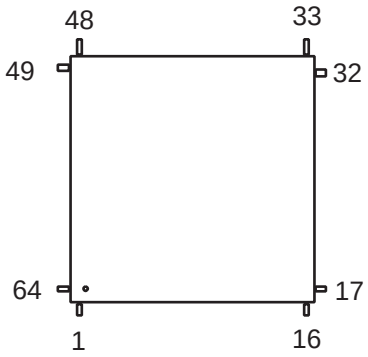
Notes:
1. WKUPx can be used if the PIO Controller defines the I/O line as "input".
2. To select this extra function, refer to Section 32.5.14 “Parallel Capture Mode”.
3. PIODCEN1/PIODCx has priority over WKUPx. Refer to Section 32.5.14 “Parallel Capture Mode”.
4. Refer to Section 22.4.2 “Slow Clock Generator”.
5. To select this extra function, refer to Section 50.5.1 “I/O Lines”
6. Analog input has priority over WKUPx pin. To select the analog input, refer to Section 50.5.1 “I/O Lines”. WKUPx can be used if the PIO controller defines the I/O line as "input".
7. Analog input has priority over RTCOUTx pin. To select the analog input, refer to Section 50.5.1 “I/O Lines”. Refer to Section 26.5.8 “Waveform Generation” to select RTCOUTx.
8. Analog input has priority over WKUPx pin. To select the analog input, refer to Section 50.5.1 “I/O Lines”. To select PIODCEN2, refer to Section 32.5.14 “Parallel Capture Mode”.
9. Refer to the System I/O Configuration Register in Section 18. “Bus Matrix (MATRIX)”.
10. Refer to Section 30.5.3 “3 to 20 MHz Crystal or Ceramic Resonator-based Oscillator”.
11. DAC0 is selected when DACC_CHER.CH0 is set. DAC1 is selected when DACC_CHER.CH1 is set. Refer to Section 51.7.4 “DACC Channel Enable Register”.

Downloaded from Arrow.com.

5.5 64-lead Package

5.5.1 64-pin LQFP Package Outline

Figure 5-6. Orientation of the 64-pin LQFP Package



Downloaded from Arrow.com.

Downloaded from Arrow.com.

5.6 64-lead Package Pinout

Table 5-3. 64-lead LQFP Package Pinout

LQFP Pin	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		PIO Peripheral D		Reset State
			Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, HiZ, ST
40	VDDIO	GPIO_AD	PA3	I/O	PIODC0[1]	I	[2]TWD0	I/O	LONCOL1	I	PCK2	O	–	–	PIO, I, PU, ST
34	VDDIO	GPIO	PA4	I/O	WKUP3/PIODC1[2]	I	TWCK0	O	TCLK0	I	UTXD1	O	–	–	PIO, I, PU, ST
32	VDDIO	GPIO_AD	PA5	I/O	WKUP4/PIODC2[2]	I	PWMC1_PWML3	O	ISI_D4	I	URXD1	I	–	–	PIO, I, PU, ST
15	VDDIO	CLOCK	PA7	I/O	XIN32[3]	I	–	–	PWMC0_PWMH3	–	–	–	–	–	PIO, HiZ
16	VDDIO	CLOCK	PA8	I/O	XOUT32[3]	O	PWMC1_PWMH3	O	AFE0_ADTRG	I	–	–	–	–	PIO, HiZ
33	VDDIO	GPIO_AD	PA9	I/O	WKUP6/PIODC3[2]	I	URXD0	I	ISI_D3	I	PWMC0_PWMFI0	I	–	–	PIO, I, PU, ST
28	VDDIO	GPIO_AD	PA10	I/O	PIODC4[1]	I	UTXD0	O	PWMC0_PWMEXTRG0	I	RD	I	–	–	PIO, I, PU, ST
27	VDDIO	GPIO_AD	PA11	I/O	WKUP7/PIODC5[2]	I	QCS	O	PWMC0_PWMH0	O	PWMC1_PWML0	O	–	–	PIO, I, PU, ST
29	VDDIO	GPIO_AD	PA12	I/O	PIODC6[1]	I	QIO1	I/O	PWMC0_PWMH1	O	PWMC1_PWMH0	O	–	–	PIO, I, PU, ST
18	VDDIO	GPIO_AD	PA13	I/O	PIODC7[1]	I	QIO0	I/O	PWMC0_PWMH2	O	PWMC1_PWML1	O	–	–	PIO, I, PU, ST
19	VDDIO	GPIO_CLK	PA14	I/O	WKUP8/PIODCEN1[2]	I	QSCK	O	PWMC0_PWMH3	O	PWMC1_PWMH1	O	–	–	PIO, I, PU, ST
12	VDDIO	GPIO_AD	PA21	I/O	AFE0_AD1/ PIODCEN2[7]	I	RXD1	I	PCK1	O	PWMC1_PWMFI0	I	–	–	PIO, I, PU, ST
17	VDDIO	GPIO_AD	PA22	I/O	PIODCCLK[1]	I	RK	I/O	PWMC0_PWMEXTRG1	I	NCS2	O	–	–	PIO, I, PU, ST
23	VDDIO	GPIO_AD	PA24	I/O	–	–	RTS1	O	PWMC0_PWMH1	O	A20	O	ISI_PCK	I	PIO, I, PU, ST
30	VDDIO	GPIO_AD	PA27	I/O	–	–	DTR1	O	TIOB2	I/O	MCDA3	I/O	ISI_D7	I	PIO, I, PU, ST
8	VDDIO	GPIO	PB0	I/O	AFE0_AD10/ RTCOUT0[6]	I	PWMC0_PWMH0	O	–	–	RXD0	I	TF	I/O	PIO, I, PU, ST
7	VDDIO	GPIO	PB1	I/O	AFE1_AD0/ RTCOUT1[6]	I	PWMC0_PWMH1	O	GTSUCOMP	O	TXD0	I/O	TK	I/O	PIO, I, PU, ST
9	VDDIO	GPIO	PB2	I/O	AFE0_AD5[4]	I	CANTX0	O	–	–	CTS0	I	SPI0_NPCS0	I/O	PIO, I, PU, ST
11	VDDIO	GPIO_AD	PB3	I/O	AFE0_AD2/WKUP12[6]	I	CANRX0	I	PCK2	O	RTS0	O	ISI_D2	I	PIO, I, PU, ST
46	VDDIO	GPIO_MLB	PB4	I/O	TDI[8]	I	TWD1	I/O	PWMC0_PWMH2	O	–	–	TXD1	I/O	PIO, I, PD, ST
47	VDDIO	GPIO_MLB	PB5	I/O	TDO/TRACESWO/ WKUP13[8]	O	TWCK1	O	PWMC0_PWML0	O	–	–	TD	O	O, PU
35	VDDIO	GPIO	PB6	I/O	SWDIO/TMS[8]	I	–	–	–	–	–	–	–	–	PIO,I,ST
39	VDDIO	GPIO	PB7	I/O	SWCLK/TCK[8]	I	–	–	–	–	–	–	–	–	PIO,I,ST
62	VDDIO	CLOCK	PB8	I/O	XOUT[9]	O	–	–	–	–	–	–	–	–	PIO, HiZ
63	VDDIO	CLOCK	PB9	I/O	XIN[9]	I	–	–	–	–	–	–	–	–	PIO, HiZ
38	VDDIO	GPIO	PB12	I/O	ERASE[8]	I	PWMC0_PWML1	O	GTSUCOMP	O	–	–	PCK0	O	PIO, I, PD, ST
1	VDDIO	GPIO_AD	PD0	I/O	DAC1[10]	I	GTXCK	I	PWMC1_PWML0	O	SPI1_NPCS1	I/O	DCD0	I	PIO, I, PU, ST
57	VDDIO	GPIO	PD1	I/O	–	–	GTXEN	O	PWMC1_PWMH0	O	SPI1_NPCS2	I/O	DTR0	O	PIO, I, PU, ST
56	VDDIO	GPIO	PD2	I/O	–	–	GTX0	O	PWMC1_PWML1	O	SPI1_NPCS3	I/O	DSR0	I	PIO, I, PU, ST
55	VDDIO	GPIO	PD3	I/O	–	–	GTX1	O	PWMC1_PWMH1	O	UTXD4	O	RI0	I	PIO, I, PU, ST

Downloaded from Arrow.com.

Table 5-3. 64-lead LQFP Package Pinout (Continued)

LQFP Pin	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		PIO Peripheral D		Reset State
			Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, HiZ, ST
54	VDDIO	GPIO_CLK	PD4	I/O	–	–	GRXDV	I	PWMC1_PWML2	O	TRACED0	O	DCD2	I	PIO, I, PU, ST
53	VDDIO	GPIO_CLK	PD5	I/O	–	–	GRX0	I	PWMC1_PWMH2	O	TRACED1	O	DTR2	O	PIO, I, PU, ST
51	VDDIO	GPIO_CLK	PD6	I/O	–	–	GRX1	I	PWMC1_PWML3	O	TRACED2	O	DSR2	I	PIO, I, PU, ST
50	VDDIO	GPIO_CLK	PD7	I/O	–	–	GRXER	I	PWMC1_PWMH3	O	TRACED3	O	RI2	I	PIO, I, PU, ST
49	VDDIO	GPIO_CLK	PD8	I/O	–	–	GMDC	O	PWMC0_PWMFI1	I	–	–	TRACECLK	O	PIO, I, PU, ST
48	VDDIO	GPIO_CLK	PD9	I/O	–	–	GMDIO	I/O	PWMC0_PWMFI2	I	AFE1_ADTRG	I	–	–	PIO, I, PU, ST
44	VDDIO	GPIO_MLB	PD10	I/O	–	–	GCRS	I	PWMC0_PWML0	O	TD	O	–	–	PIO, I, PD, ST
43	VDDIO	GPIO_AD	PD11	I/O	–	–	GRX2	I	PWMC0_PWMH0	O	GTSUCOMP	O	ISI_D5	I	PIO, I, PU, ST
41	VDDIO	GPIO_AD	PD12	I/O	–	–	GRX3	I	CANTX1	O	SPI0_NPCS2	O	ISI_D6	I	PIO, I, PU, ST
26	VDDIO	GPIO_AD	PD21	I/O	–	–	PWMC0_PWMH1	O	SPI0_MOSI	I/O	TIOA11	I/O	ISI_D1	I	PIO, I, PU, ST
25	VDDIO	GPIO_AD	PD22	I/O	–	–	PWMC0_PWMH2	O	SPI0_SPCK	O	TIOB11	I/O	ISI_D0	I	PIO, I, PU, ST
22	VDDIO	GPIO_AD	PD24	I/O	–	–	PWMC0_PWML0	O	RF	I/O	TCLK11	I	ISI_HSYNC	I	PIO, I, PU, ST
20	VDDIO	GPIO_AD	PD25	I/O	–	–	PWMC0_PWML1	O	SPI0_NPCS1	I/O	URXD2	I	ISI_VSYNC	I	PIO, I, PU, ST
21	VDDIO	GPIO	PD26	I/O	–	–	PWMC0_PWML2	O	TD	O	UTXD2	O	UTXD1	O	PIO, I, PU, ST
2	VDDIO	GPIO_AD	PD31	I/O	–	–	QIO3	I/O	UTXD3	O	PCK2	O	ISI_D11	I	PIO, I, PU, ST
3	VDDOUT	Power	VDDOUT	–	–	–	–	–	–	–	–	–	–	–	–
4	VDDIN	Power	VDDIN	–	–	–	–	–	–	–	–	–	–	–	–
5	VDDIO	Reference	VREFP	I	–	–	–	–	–	–	–	–	–	–	–
36	VDDIO	RST	NRST	I/O	–	–	–	–	–	–	–	–	–	–	PIO, I, PU
37	VDDIO	TEST	TST	I	–	–	–	–	–	–	–	–	–	–	I, PD
10, 42, 58	VDDIO	Power	VDDIO	–	–	–	–	–	–	–	–	–	–	–	–
45	VDDIO	TEST	JTAGSEL	I	–	–	–	–	–	–	–	–	–	–	I, PD
13, 24, 61	VDDCORE	Power	VDDCORE	–	–	–	–	–	–	–	–	–	–	–	–
52	VDDPLL	Power	VDDPLL	–	–	–	–	–	–	–	–	–	–	–	–
59	VDDUTMII	USBHS	DM	I/O	–	–	–	–	–	–	–	–	–	–	–
60	VDDUTMII	USBHS	DP	I/O	–	–	–	–	–	–	–	–	–	–	–
6, 14, 31	GND	Ground	GND	–	–	–	–	–	–	–	–	–	–	–	–
64	VDDPLLUSB	Power	VDDPLLUSB	–	–	–	–	–	–	–	–	–	–	–	–

Notes:
1. To select this extra function, refer to Section 32.5.14 “Parallel Capture Mode”.
2. PIODCEN1/PIODCx has priority over WKUPx. Refer to Section 32.5.14 “Parallel Capture Mode”.
3. Refer to Section 22.4.2 “Slow Clock Generator”.
4. To select this extra function, refer to Section 50.5.1 “I/O Lines”.
5. Analog input has priority over WKUPx pin. To select the analog input, refer to Section 50.5.1 “I/O Lines”. WKUPx can be used if the PIO controller defines the I/O line as "input".
6. Analog input has priority over RTCOUTx pin. To select the analog input, refer to Section 50.5.1 “I/O Lines”. Refer to Section 26.5.8 “Waveform Generation” to select RTCOUTx.
7. Analog input has priority over WKUPx pin. To select the analog input, refer to Section 50.5.1 “I/O Lines”. To select PIODCEN2, refer to Section 32.5.14 “Parallel Capture Mode”.
8. Refer to the System I/O Configuration Register in Section 18. “Bus Matrix (MATRIX)”.
9. Refer to Section 30.5.3 “3 to 20 MHz Crystal or Ceramic Resonator-based Oscillator”.
10. DAC0 is selected when DACC_CHER.CH0 is set. DAC1 is selected when DACC_CHER.CH1 is set. Refer to Section 51.7.4 “DACC Channel Enable Register”.

Downloaded from Arrow.com.

6. Power Considerations

6.1 Power Supplies

Table 6-1 defines the power supply rails of the SAM E70 and the estimated power consumption at typical voltage.

Table 6-1. Power Supplies

Name	Associated Ground	Powers
VDDCORE	GND	Core, embedded memories and peripherals
VDDIO	GND	Peripheral I/O lines (Input/Output Buffers), backup part, 1 Kbytes of backup SRAM, 32 kHz crystal oscillator, oscillator pads. For USB operations, VDDIO voltage range must be between 3.0V and 3.6V.
VDDIN	GND, GNDANA	Voltage regulator input. Supplies also the ADC, DAC and analog voltage comparator.
VDDPLL	GND, GNDPLL	PLLA and the fast RC oscillator
VDDPLLUSB	GND, GNDPLLUSB	UTMI PLL and the 3 to 20 MHz oscillator. For USB operations, VDDPLLUSB must be between 3.0V and 3.6V.
VDDUTMII	GNDUTMI	USB transceiver interface. Must be connected to VDDIO. For USB operations, VDDUTMII voltage range must be between 3.0V and 3.6V.
VDDUTMIC	GNDUTMI	USB transceiver core

6.2 Power Constraints

The following power constraints apply to SAM E70 devices. Deviating from these constraints may lead to unpredictable results.

- VDDIN and VDDIO must have the same level
- VDDIN and VDDIO must always be higher than or equal to VDDCORE
- VDDCORE, VDDPLL and VDDUTMIC voltage levels must not vary by more than 0.6V.
- For the USB to be operational, VDDUTMII, VDDPLLUSB, VDDIN and VDDIO must be higher than or equal to 3.0V.

Downloaded from Arrow.com.

6.2.1 Power-up

VDDIO and VDDIN must rise simultaneously, prior to VDDCORE, VDDPLL and VDDUTMIC rising. This is respected if VDDCORE, VDDPLL and VDDUTMIC are supplied by the embedded voltage regulator.

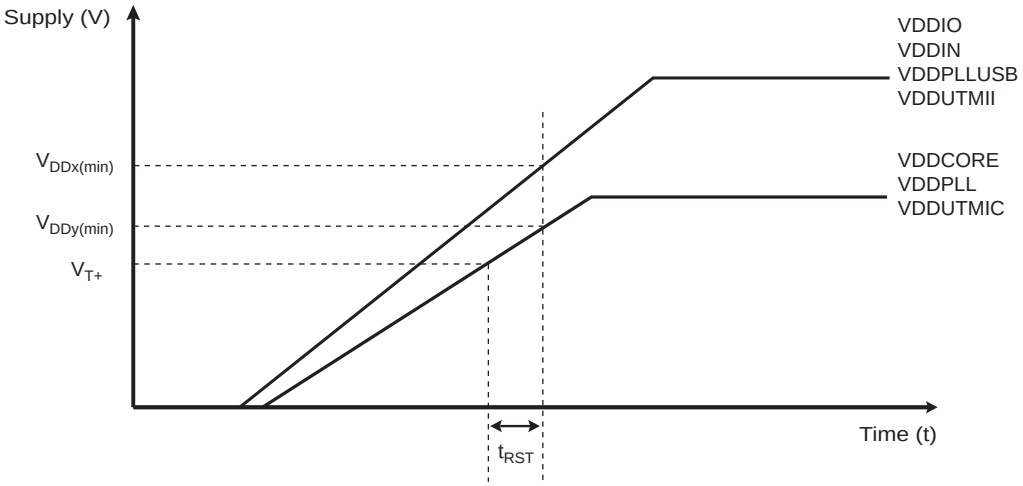
If VDDCORE is powered by an external voltage regulator, VDDIO and VDDIN must reach their minimum operating voltage before $VDDCORE_{min}$. The minimum slope for VDDCORE is defined by:

$$(VDDCORE_{min} - V_{T+})/(t_{RES})$$

If VDDCORE rises at the same time as VDDIO and VDDIN, the rising slope of VDDIO and VDDIN must be higher than or equal to 2.4V/ms. Refer to Table 56-9 “VDDIO Power-On Reset Characteristics“.

In order to prevent any overcurrent at power-up, it is required that ADVREFP rises simultaneously with VDDIO and VDDIN.

Figure 6-1. Power-up Sequence



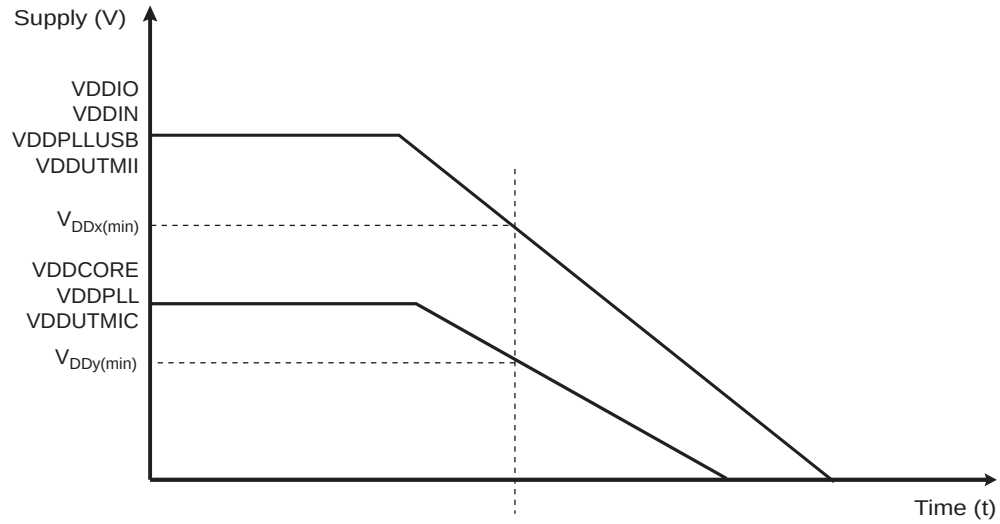
Downloaded from Arrow.com.

6.2.2 Power-down

If VDDCORE, VDDPLL and VDDUTMIC are not supplied by the embedded voltage regulator, VDDIO, VDDIN, VDDPLLUSB and VDDUTMII should fall simultaneously, prior to VDDCORE, VDDPLL and VDDUTMIC falling. The VDDCORE falling slope must not be faster than 20V/ms.

In order to prevent any overcurrent at power-down, it is required that ADVREFP falls simultaneously with VDDIO and VDDIN.

Figure 6-2. Power-down Sequence



6.3 Voltage Regulator

The SAM E70 embeds a voltage regulator that is managed by the Supply Controller.

For adequate input and output power supply decoupling/bypassing, refer to Table 56-4 “1.2V Voltage Regulator Characteristics“.

6.4 Backup SRAM Power Switch

The SAM E70 embeds a power switch to supply the 1 Kbyte of backup SRAM. It is activated only when VDDCORE is switched off to ensure retention of the contents of the backup SRAM. When VDDCORE is switched on, the backup SRAM is powered with VDDCORE.

To save the power consumption of the backup SRAM, the user can disable the backup SRAM power switch by clearing the bit SRAMON in the Supply Controller Mode Register (SUPC_MR). By default, after VDDIO rises, the backup SRAM power switch is enabled.

6.5 Active Mode

Active mode is the normal running mode with the core clock running from the fast RC oscillator, the main crystal oscillator or the PLLA. The Power Management Controller can be used to adapt the core, bus and peripheral frequencies and to enable and/or disable the peripheral clocks.

Downloaded from Arrow.com.

6.6 Low-power Modes

The SAM E70 features low-power modes:

- Backup mode
- Wait mode
- Sleep mode

6.6.1 Backup Mode

The purpose of Backup mode is to achieve the lowest power consumption possible in a system which is performing periodic wake-ups to perform tasks but not requiring fast startup time.

The Supply Controller, zero-power power-on reset, RTT, RTC, backup SRAM, backup registers and 32 kHz oscillator (RC or crystal oscillator selected by software in the Supply Controller) are running. The regulator and the core supply are off.

Backup mode is based on the Cortex-M7 Deep Sleep mode with the voltage regulator disabled.

Wake-up from Backup mode is done through WKUP0–13 pins, the supply monitor (SM), the RTT, or an RTC wake-up event.

Backup mode is entered by using bit VROFF in the Supply Controller Control Register (SUPC_CR) and the SLEEPDEEP bit in the Cortex-M7 System Control Register set to 1. Refer to information on Power Management in the ARM Cortex-M7 documentation available at www.arm.com.

To enter Backup mode, follow the steps below:

1. Set the SLEEPDEEP bit of the Cortex-M7 processor.
2. Set the VROFF bit of SUPC_CR.

Exit from Backup mode occurs as a result of one of the following enabled wake-up events:

- WKUP0–13 pins (level transition, configurable debouncing)
- Supply Monitor alarm
- RTC alarm
- RTT alarm

6.6.2 Wait Mode

The purpose of Wait mode is to achieve very low power consumption while maintaining the whole device in a powered state for a startup time of less than 10 µs.

In Wait mode, the clocks of the core, peripherals and memories are stopped. However, the core, peripherals and memories power supplies are still powered.

Wait mode is entered when the bit WAITMODE is set in CKGR_MOR and the field FLPM is configured to 00 or 01 in the PMC Fast Startup Mode register (PMC_FSMR).

The Cortex-M is able to handle external events or internal events in order to wake up the core. This is done by configuring the external lines WKUP0–13 as fast startup wake-up pins (refer to Section 6.8 “Fast Startup”). RTC or RTT alarms or USB wake-up events can be used to wake up the processor. Resume from Wait mode is also achieved when a debug request occurs and the bit CDBGPWRUPREQ is set in the processor.

To enter Wait mode, follow the steps below:

1. Select the 4/8/12 MHz fast RC oscillator as Main Clock.
2. Configure the FLPM field in the PMC_FSMR.
3. Set Flash Wait State at 0.
4. Set HCLK = MCK by configuring MDIV to 0 in the PMC Master Clock register (PMC_MCKR).
5. Set the WAITMODE bit in the PMC Clock Generator Main Oscillator register (CKGR_MOR).
6. Wait for MCKRDY = 1 in the PMC Status register (PMC_SR).

Downloaded from Arrow.com.

Note: Internal main clock resynchronization cycles are necessary between writing the MOSCRCEN bit and the entry in Wait mode. Depending on the user application, waiting for MOSCRCEN bit to be cleared is recommended to ensure that the core will not execute undesired instructions.

6.6.3 Sleep Mode

The purpose of sleep mode is to optimize power consumption of the device versus response time. In this mode, only the core clock is stopped. The peripheral clocks can be enabled. The current consumption in this mode is application-dependent.

This mode is entered using the instruction Wait for Interrupt (WFI).

Processor wake-up is triggered by an interrupt if the WFI instruction of the Cortex-M processor is used.

Downloaded from Arrow.com.

Downloaded from Arrow.com.

6.6.4 Low-Power Mode Summary Table

The modes detailed above are the main low-power modes. Each part can be set to on or off separately and wake up sources can be individually configured. Table 6-2 below shows a summary of the configurations of the low-power modes.

Table 6-2. Low-power Mode Configuration Summary

Mode	SUPC, 32 kHz Oscillator, RTC, RTT Backup SRAM (BRAM), Backup Registers (GPBR), POR (Backup Area)	Regulator	Core Memory Peripherals	Mode Entry Configuration	Potential Wake-Up Sources	Core at Wake-Up	PIO State while in Low Power Mode	PIO State at Wake-Up	Wake-up Time(2)
Backup Mode	ON	OFF	OFF (Not powered)	SUPC_CR.VROFF = 1 SLEEPDEEP(1) = 1	WKUP0–13 pins Supply Monitor RTC alarm RTT alarm	Reset	Previous state maintained	PIOA, PIOB, PIOC, PIOD & PIOE inputs with pull-ups	< 2 ms
Wait Mode w/Flash in Deep Power-down Mode	ON	ON	Powered (Not clocked)	PMC_MCKR.MDIV = 0 CKGR_MOR.WAITMODE =1 SLEEPDEEP(1) = 0 PMC_FSMR.LPM = 1 PMC_FSMR.FLPM = 1	WKUP0–13 pins RTC RTT USBHS Processor debug(6) GMAC Wake on LAN event Wake-up from CAN (7)	Clocked back(3)	Previous state maintained	Unchanged	< 10 µs
Wait Mode w/Flash in Standby Mode	ON	ON	Powered (Not clocked)	PMC_MCKR.MDIV = 0 CKGR_MOR.WAITMODE =1 SLEEPDEEP(1) = 0 PMC_FSMR.LPM = 1 PMC_FSMR.FLPM = 0	WKUP0–13 pins RTC RTT USBHS Processor debug(6) GMAC Wake on LAN Wake-up from CAN (7)	Clocked back(3)	Previous state maintained	Unchanged	< 10 µs
Sleep Mode	ON	ON	Powered(4) (Not clocked)	WFI SLEEPDEEP(1) = 0 PMC_FSMR.LPM = 0	Any enabled Interrupt	Clocked back	Previous state maintained	Unchanged	(5)

Notes:
1. The bit SLEEPDEEP is in the Cortex-M7 System Control Register.
2. When considering wake-up time, the time required to start the PLL is not taken into account. Once started, the device works with the 4/8/12 MHz fast RC oscillator. The user has to add the PLL start-up time if it is needed in the system. The wake-up time is defined as the time taken for wake up until the first instruction is fetched.
3. HCLK = MCK. The user may need to revert back to the previous clock configuration.
4. Depends on MCK frequency.
5. In this mode, the core is supplied and not clocked. Some peripherals can be clocked.
6. Resume from Wait mode if a debug request occurs (CDBGPWRUPREQ is set in the processor).
7. CAN wakeup requires the use of any WKUP0–13 pin.

6.7 Wake-up Sources

Wake-up events allow the device to exit Backup mode. When a wake-up event is detected, the Supply Controller performs a sequence which automatically reenables the core power supply and the SRAM power supply, if they are not already enabled.

6.8 Fast Startup

The SAM E70 allows the processor to restart in a few microseconds while the processor is in Wait mode or in Sleep mode. A fast startup can occur upon detection of a low level on any of the following wake-up sources:

- WKUP0 to WKUP13 pins
- Supply Monitor
- RTC alarm
- RTT alarm
- USBHS interrupt line (WAKEUP)
- Processor debug request (CDBGPWRUPREQ)
- GMAC wake on LAN event

Note: CAN wakeup requires the use of any WKUP0–13 pin.

The fast restart circuitry is fully asynchronous and provides a fast start-up signal to the Power Management Controller. As soon as the fast start-up signal is asserted, the PMC automatically restarts the embedded 4/8/12 MHz Fast RC oscillator, switches the master clock on this 4 MHz clock and re-enables the processor clock.

Downloaded from Arrow.com.

7. Input/Output Lines

The SAM E70 features both general purpose I/Os (GPIO) and system I/Os. GPIOs can have alternate functionality due to multiplexing capabilities of the PIO controllers. The same PIO line can be used, whether in I/O mode or by the multiplexed peripherals. System I/Os include pins such as test pins, oscillators, erase or analog inputs.

7.1 General-Purpose I/O Lines

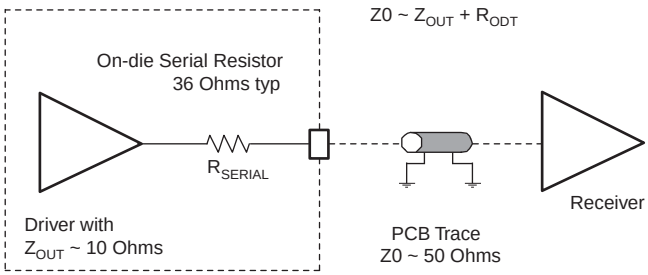
General-purpose (GPIO) lines are managed by PIO Controllers. All I/Os have several input or output modes such as pull-up or pull-down, input Schmitt triggers, multi-drive (open-drain), glitch filters, debouncing or input change interrupt. Programming of these modes is performed independently for each I/O line through the PIO controller user interface. For more details, refer to Section 32. “Parallel Input/Output Controller (PIO)”.

The input/output buffers of the PIO lines are supplied through VDDIO power supply rail.

The SAM E70 embeds high-speed pads able to handle the high-speed clocks for HSMCI, SPI and QSPI (MCK/2). Refer to the Section 56. “Electrical Characteristics” for more details. Typical pull-up and pull-down value is 100 kΩ for all I/Os.

Each I/O line also embeds an R_{SERIAL} (On-die Serial Resistor), (see Figure 7-1 below). It consists of an internal series resistor termination scheme for impedance matching between the driver output (SAM E70) and the PCB trace impedance preventing signal reflection. The series resistor helps to reduce IOs switching current (di/dt) thereby reducing in turn, EMI. It also decreases overshoot and undershoot (ringing) due to inductance of interconnect between devices or between boards. Finally, R_{SERIAL} helps diminish signal integrity issues.

Figure 7-1. On-Die Termination



Downloaded from Arrow.com.

7.2 System I/O Lines

System I/O lines are pins used by oscillators, test mode, reset, JTAG and other features. Table 7-1 lists the SAM E70 system I/O lines shared with PIO lines.

These pins are software-configurable as general-purpose I/Os or system pins. At startup, the default function of these pins is always used.

Table 7-1. System I/O Configuration Pin List.

CCFG_SYSIO Bit Number	Default Function After Reset	Other Function	Constraints for Normal Start	Configuration
12	ERASE	PB12	Low Level at startup(1)	In Matrix User Interface Registers (Refer to the System I/O Configuration Register in Section 18. "Bus Matrix (MATRIX)")
7	TCK/SWCLK	PB7	–	
6	TMS/SWDIO	PB6	–	
5	TDO/TRACESWO	PB5	–	
4	TDI	PB4	–	
–	PA7	XIN32	–	(2)
–	PA8	XOUT32	–	
–	PB9	XIN	–	(3)
–	PB8	XOUT	–	

Notes:
1. If PB12 is used as PIO input in user applications, a low level must be ensured at startup to prevent Flash erase before the user application sets PB12 into PIO mode,
2. Refer to Section 22.4.2 "Slow Clock Generator".
3. Refer to Section 30.5.3 "3 to 20 MHz Crystal or Ceramic Resonator-based Oscillator".

7.2.1 Serial Wire Debug Port (SW-DP) Pins

The SW-DP pins SWCLK and SWDIO are commonly provided on a standard 20-pin JTAG connector defined by ARM. For more details about voltage reference and reset state, refer to Table 4-1 "Signal Description List".

At startup, SW-DP pins are configured in SW-DP mode to allow connection with debugging probe. For more details, refer to Section 15. "Debug and Test Features".

SW-DP pins can be used as standard I/Os to provide users more general input/output pins when the debug port is not needed in the end application. Mode selection between SW-DP mode (System IO mode) and general IO mode is performed through the AHB Matrix Special Function Registers (MATRIX_SFR). Configuration of the pad for pull-up, triggers, debouncing and glitch filters is possible regardless of the mode.

The JTAGSEL pin is used to select the JTAG boundary scan when asserted at a high level. It integrates a permanent pull-down resistor of about 15 kΩ to GND, so that it can be left unconnected for normal operations.

The JTAG Debug Port TDI, TDO, TMS and TCK is inactive. It is provided for Boundary Scan Manufacturing Test purpose only.

7.2.2 Embedded Trace Module (ETM) Pins

The Embedded Trace Module (ETM) depends on the Trace Port Interface Unit (TPIU) to export data out of the system.

The TPUI features the following pins:

- TRACECLK is always exported to enable synchronization with the data.
- TRACED0–TRACED3 is the instruction trace stream.

Downloaded from Arrow.com.

7.3 NRST Pin

The NRST pin is bidirectional. It is handled by the on-chip Reset Controller (RSTC) and can be driven low to provide a reset signal to the external components or asserted low externally to reset the microcontroller. It resets the core and the peripherals, with the exception of the Backup area (RTC, RTT, Backup SRAM and Supply Controller). The NRST pin integrates a permanent pull-up resistor to VDDIO of about 100 kΩ.

By default, the pin is configured as an input.

7.4 ERASE Pin

The ERASE pin is used to reinitialize the Flash content and some of its NVM bits to an erased state (all bits read as logic level 1). The ERASE pin and the ROM code ensure an in-situ reprogrammability of the Flash content without the use of a debug tool. When the security bit is activated, the ERASE pin provides the capability to reprogram the Flash content. The ERASE pin integrates a pull-down resistor of about 100 kΩ to GND, so that it can be left unconnected for normal operations.

This pin is debounced by SLCK to improve the glitch tolerance. To avoid unexpected erase at power-up, a minimum ERASE pin assertion time is required. This time is defined in Table 56-52 “Flash Characteristics“.

The ERASE pin is a system I/O pin that can be used as a standard I/O. At startup, this system I/O pin defaults to the ERASE function. To avoid unexpected erase at power-up due to glitches, a minimum ERASE pin assertion time is required. This time is defined in Table 56-52 “Flash Characteristics“.

The erase operation cannot be performed when the system is in Wait mode.

If the ERASE pin is used as a standard I/O in Input or Output mode, note the following considerations and behavior:

- I/O Input mode: at startup of the device, the logic level of the pin must be low to prevent unwanted erasing until the user application has reconfigured this system I/O pin to a standard I/O pin.
- I/O Output mode: asserting the pin to low does not erase the Flash

During software application development, a faulty software may put the device into a deadlock. This may be due to:

- programming an incorrect clock switching sequence
- using this system I/O pin as a standard I/O pin
- entering Wait mode without any wake-up events programmed

The only way to recover normal behavior is to erase the Flash by following the steps below:

1. Apply a logic "1" level on the ERASE pin.
2. Apply a logic "0" level on the NRST pin.
3. Power-down then power-up the device.
4. Maintain the ERASE pin to logic "0" level for at least the minimum assertion time after releasing the NRST pin to logic "1" level.

Downloaded from Arrow.com.

8. Interconnect

The system architecture is based on the ARM Cortex-M7 processor connected to the main AHB Bus Matrix, the embedded Flash, the multi-port SRAM and the ROM.

The 32-bit AHBP interface is a single 32-bit wide interface that accesses the peripherals connected on the main Bus Matrix. It is used only for data access. Instruction fetches are never performed on the AHBP interface. The bus, AHBP or AXIM, accessing the peripheral memory area [0x40000000 to 0x60000000] is selected in the AHBP control register.

The 32-bit AHBS interface provides system access to the ITCM, D1TCM, and D0TCM. It is connected on the main Bus Matrix and allows the XDMA to transfer from memory or peripherals to the instruction or data TCMs.

The 64-bit AXIM interface is a single 64-bit wide interface connected through two ports of the AXI Bridge to the main AHB Bus Matrix and to two ports of the multi-port SRAM. The AXIM interface allows:

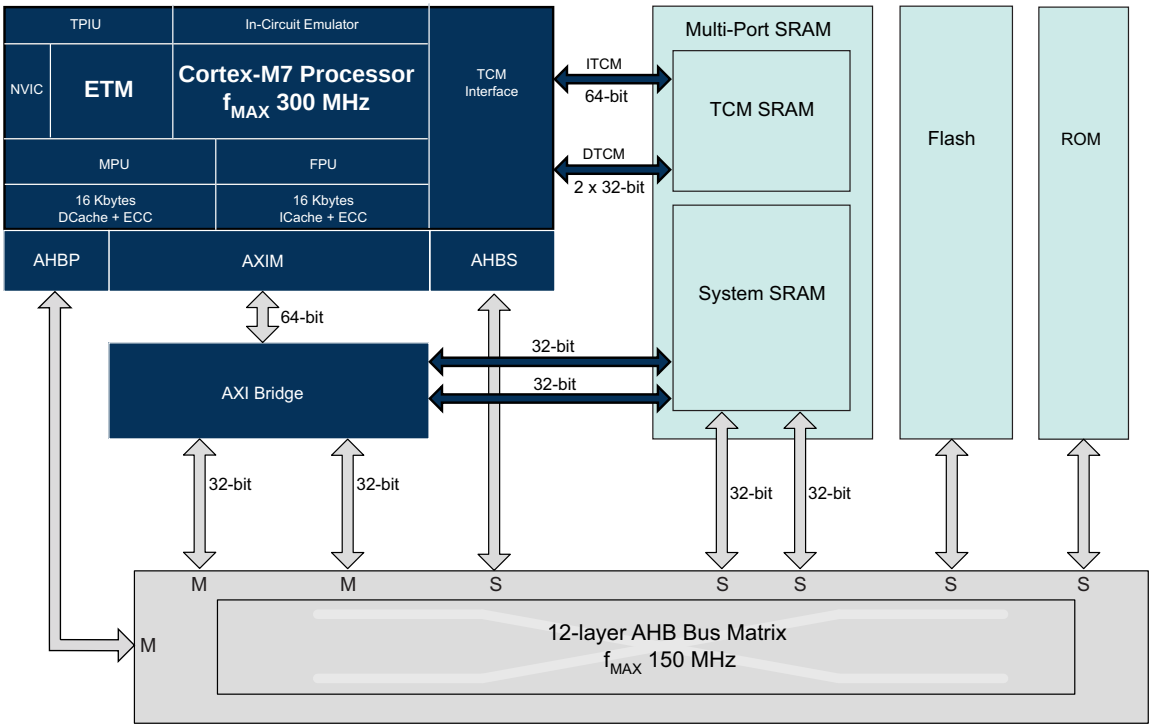
- Instruction fetches
- Data cache linefills and evictions
- Non-cacheable normal-type memory data accesses
- Device and strongly-ordered type data accesses, generally to peripherals

The interleaved multi-port SRAM optimizes the Cortex-M7 accesses to the internal SRAM.

The interconnect of the other masters and slaves is described in Section 18. “Bus Matrix (MATRIX)”.

Figure 8-1 shows the connections of the different Cortex-M7 ports.

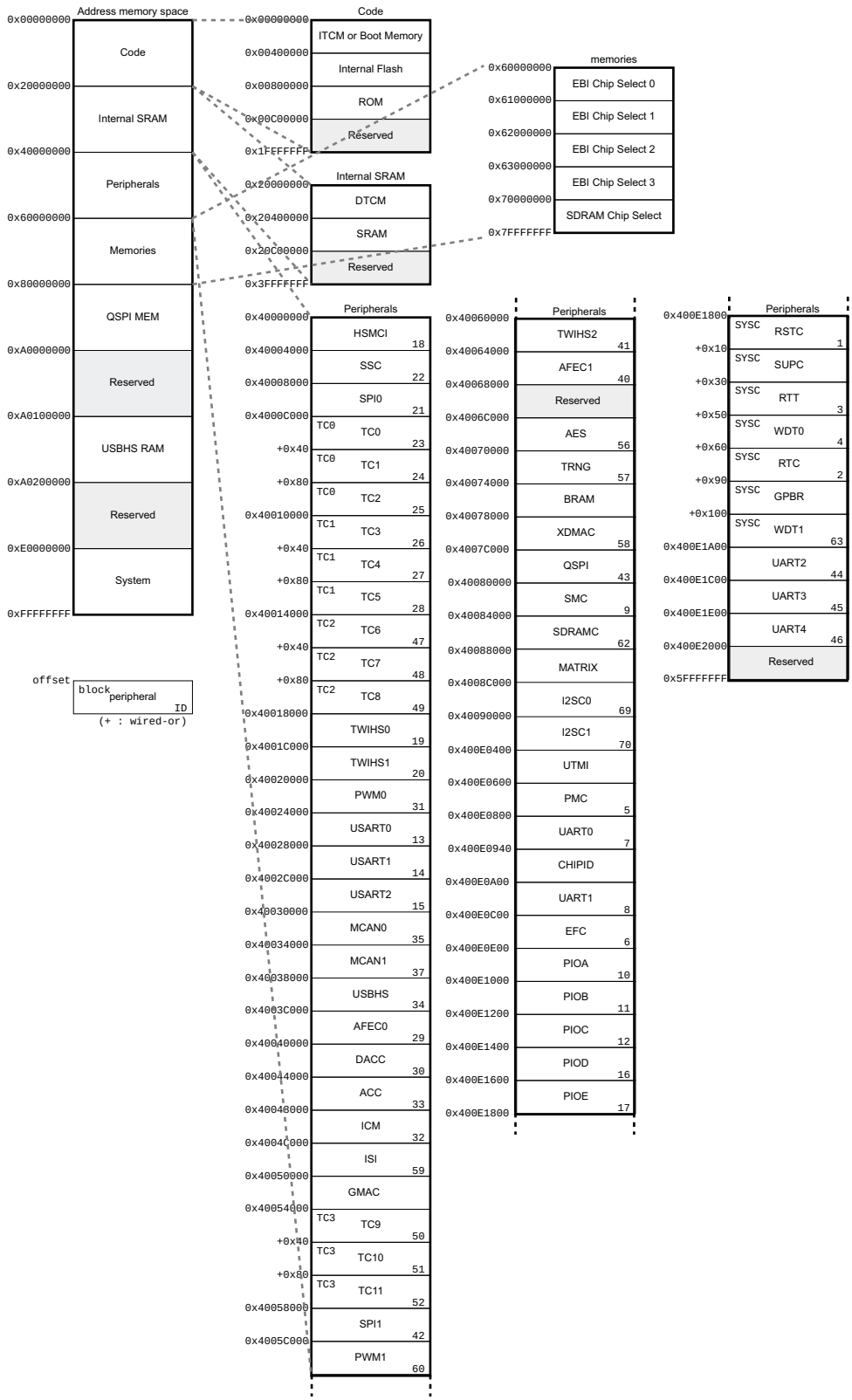
Figure 8-1. Interconnect Block Diagram



Downloaded from Arrow.com.

9. Product Mapping

Figure 9-1. SAM E70 Product Mapping



Downloaded from Arrow.com.

10. Memories

10.1 Embedded Memories

10.1.1 Internal SRAM

SAM E70 devices embed 384 Kbytes or 256 Kbytes of high-speed SRAM.

The SRAM is accessible over the system Cortex-M bus at address 0x2040 0000.

SAM E70 devices embed a Multi-Port SRAM with four ports to optimize the bandwidth and latency. The priorities, defined in the Bus Matrix for each SRAM port slave are propagated, for each request, up to the SRAM slaves.

The Bus Matrix supports four priority levels: Normal, Bandwidth-sensitive, Latency-sensitive and Latency-critical in order to increase the overall processor performance while securing the high-priority latency-critical requests from the peripherals.

The SRAM controller manages interleaved addressing of SRAM blocks to minimize access latencies. It uses Bus Matrix priorities to give the priority to the most urgent request. The less urgent request is performed no later than the next cycle.

Two SRAM slave ports are dedicated to the Cortex-M7 while two ports are shared by the AHB masters.

10.1.2 Tightly Coupled Memory (TCM) Interface

SAM E70 devices embed Tightly Coupled Memory (TCM) running at processor speed.

- ITCM is a single 64-bit interface, based at 0x0000 0000 (code region).
- DTCM is composed of dual 32-bit interfaces interleaved, based at 0x2000 0000 (data region).

ICTM and DTCM are enabled/disabled in the ITCMR and DTCMR registers in ARM SCB.

DTCM is enabled by default at reset. ITCM is disabled by default at reset.

There are four TCM configurations controlled by software. When enabled, ITCM is located at 0x0000 0000, overlapping ROM or Flash depending on the general-purpose NVM bit 1 (GPNVM). The configuration is done with GPNVM bits [8:7].

Table 10-1. TCM Configurations in Kbytes

ITCM	DTCM	SRAM for 384K RAM-based	SRAM for 256K RAM-based	GPNVM Bits [8:7]
0	0	384	256	0
32	32	320	192	1
64	64	256	128	2
128	128	128	0	3

Accesses made to TCM regions when the relevant TCM is disabled and accesses made to the Code and SRAM region above the TCM size limit are performed on the AHB matrix, i.e., on internal Flash or on ROM depending on remap GPNVM bit.

Accesses made to the SRAM above the size limit will not generate aborts.

The Memory Protection Unit (MPU) can to be used to protect these areas.

10.1.3 Internal ROM

The SAM E70 embeds an Internal ROM for the SAM Boot Assistant (SAM-BA®), In Application Programming functions (IAP) and Fast Flash Programming Interface (FFPI).

At any time, the ROM is mapped at address 0x0080 0000.

The ROM may also be mapped at 0x00000000 depending on GPNVM bit setting and ITCM use.

Downloaded from Arrow.com.

10.1.4 Backup SRAM

The SAM E70 embeds 1 Kbytes of backup SRAM located at 0x4007 4000.

The backup SRAM is accessible in 32-bit words only. Byte or half-word accesses are not supported.

The backup SRAM is supplied by VDDCORE in Normal mode.

In Backup mode, the backup SRAM supply is automatically switched to VDDIO through the backup SRAM power switch when VDDCORE falls. For more details, see Section 6.4 “Backup SRAM Power Switch”.

10.1.5 Flash Memories

SAM E70 devices embed 512 Kbytes, 1024 Kbytes or 2 Mbytes of internal Flash mapped at address 0x40 0000.

The devices feature a Quad SPI (QSPI) interface, mapped at address 0x80000000, that extends the Flash size by adding an external SPI or QSPI Flash.

When accessed by the Cortex-M7 processor for programming operations, the QSPI and internal Flash address spaces must be defined in the Cortex-M7 memory protection unit (MPU) with the attribute 'Device' or 'Strongly Ordered'. For fetch or read operations, the attribute ‘Normal memory’ must be set to benefit from the internal cache. Refer to the ARM Cortex-M7 Technical Reference Manual (ARM DDI 0489) available on www.arm.com.

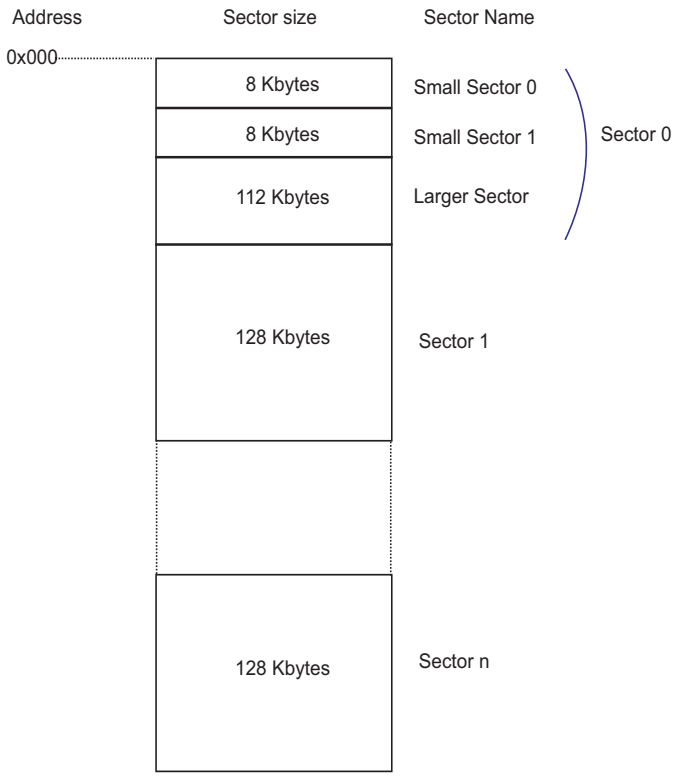
Some precautions must be taken when the accesses are performed by the central DMA. Refer to Section 21. “Enhanced Embedded Flash Controller (EEFC)” and Section 41. “Quad SPI Interface (QSPI)”.

10.1.5.1 Embedded Flash Overview

The memory is organized in sectors. Each sector has a size of 128 Kbytes. The first sector is divided into 3 smaller sectors.

The three smaller sectors are organized in 2 sectors of 8 Kbytes and 1 sector of 112 Kbytes. Refer to Figure 10-1 below.

Figure 10-1. Global Flash Organization



Each sector is organized in pages of 512 bytes.

For sector 0:

Downloaded from Arrow.com.

- The smaller sector 0 has 16 pages of 512 bytes
- The smaller sector 1 has 16 pages of 512 bytes
- The larger sector has 224 pages of 512 bytes

The rest of the array is composed of 128-Kbyte sectors of 256 pages of 512 bytes each. Refer to Figure 10-2 below.

Figure 10-2. Flash Sector Organization

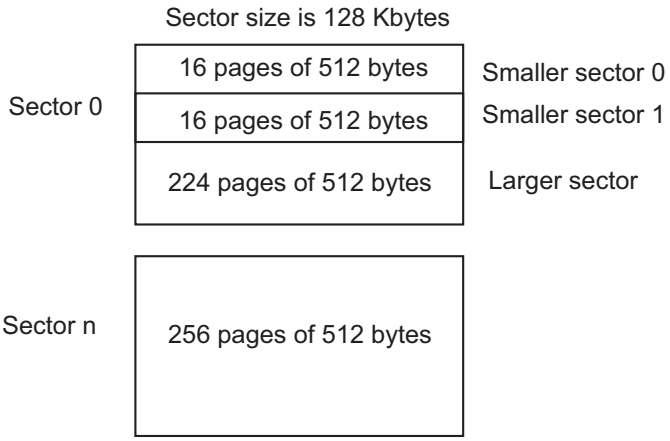
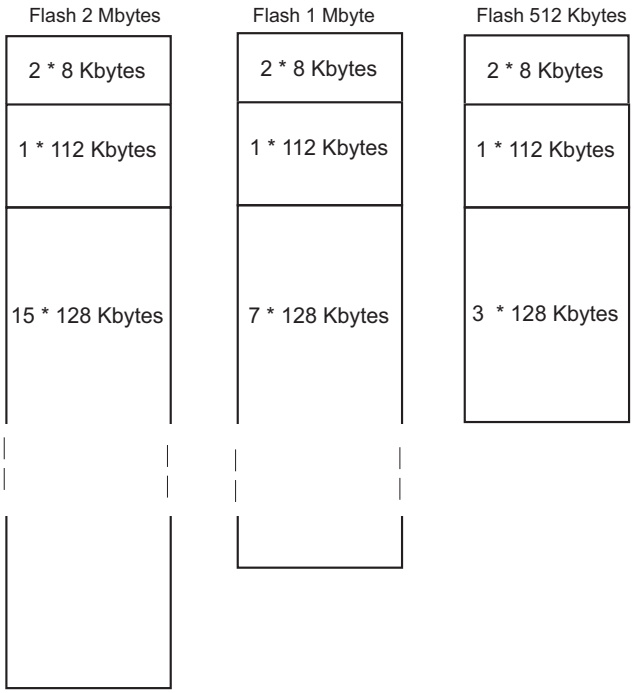


Figure 10-3 illustrates the organization of the Flash depending on its size.

Figure 10-3. Flash Size



Erasing the memory can be performed:

- by block of 8 Kbytes
- by sector of 128 Kbytes
- by 512-byte page for up to 8 Kbytes within a specific small sector
- Chip Erase

Downloaded from Arrow.com.

The memory has one additional reprogrammable page that can be used as page signature by the user. It is accessible through specific modes, for erase, write and read operations. Erase pin assertion will not erase the User Signature page.

Erase memory by page is possible only in a sector of 8 Kbytes.

EWP and EWPL commands can be only used in 8-Kbyte sectors.

10.1.5.2 Enhanced Embedded Flash Controller

Each Enhanced Embedded Flash Controller manages accesses performed by the masters of the system. It enables reading the Flash and writing the write buffer. It also contains a User Interface, mapped on the APB.

The Enhanced Embedded Flash Controller ensures the interface of the Flash block.

It manages the programming, erasing, locking and unlocking sequences of the Flash using a full set of commands.

One of the commands returns the embedded Flash descriptor definition that informs the system about the Flash organization, thus making the software generic.

10.1.5.3 Flash Speed

The user must set the number of wait states depending on the system frequency.

For more details, refer to the Embedded Flash Characteristics in Section 56. “Electrical Characteristics”.

10.1.5.4 Lock Regions

Several lock bits are used to protect write and erase operations on lock regions. A lock region is composed of several consecutive pages, and each lock region has its associated lock bit.

Table 10-2. Flash Lock Bits

Flash Size (Kbytes)	Number of Lock Bits	Lock Region Size
2048	128	16 Kbytes
1024	64	16 Kbytes
512	32	16 Kbytes

Asserting the ERASE pin clears the lock bits, thus unlocking the entire Flash.

10.1.5.5 Security Bit Feature

The SAM E70 features a security bit based on the GPNVM bit 0. When security is enabled, any access to the Flash, SRAM, core registers and internal peripherals, either through the SW-DP, the ETM interface or the Fast Flash Programming Interface, is blocked. This ensures the confidentiality of the code programmed in the Flash.

This security bit can only be enabled through the command “Set General-purpose NVM Bit 0” of the EEFC User Interface. Disabling the security bit can only be achieved by asserting the ERASE pin at 1, and after a full Flash erase is performed. When the security bit is deactivated, all accesses to the Flash, SRAM, Core registers, Internal Peripherals are permitted.

10.1.5.6 Unique Identifier

The device contains a unique identifier of 2 pages of 512 bytes. These 2 pages are read-only and cannot be erased even by the ERASE pin.

The sequence to read the unique identifier area is described in Section 21.4.3.8 “Unique Identifier Area”.

The mapping is as follows:

- Bytes [0..15]: 128 bits for unique identifier
- Bytes[16..1023]: Reserved

10.1.5.7 User Signature

Each device contains a user signature of 512 bytes that is available to the user. The user signature can be used to store information such as trimming, keys, etc., that the user does not want to be erased by asserting the ERASE pin or by software ERASE command. Read, write and erase of this area is allowed.

Downloaded from Arrow.com.

10.1.5.8 Fast Flash Programming Interface

The Fast Flash Programming Interface allows programming the device through a multiplexed fully-handshaked parallel port. It allows gang programming with market-standard industrial programmers.

The FFPI supports read, page program, page erase, full erase, lock, unlock and protect commands.

The Fast Flash Programming Interface is enabled and the Fast Programming Mode is entered when TST and PA3 and PA4 are tied low.

Table 10-3. FFPI on PIO Controller A (PIOA)

I/O Line	System Function
PD10	PGMEN0
PD11	PGMEN1
PB0	PGMM0
PB1	PGMM1
PB2	PGMM2
PB3	PGMM3
PA3	PGMNCMD
PA4	PGMRDY
PA5	PGMNOE
PA21	PGMNVALID
PA7	PGMD0
PA8	PGMD1
PA9	PGMD2
PA10	PGMD3
PA11	PGMD4
PA12	PGMD5
PA13	PGMD6
PA14	PGMD7
PD0	PGMD8
PD1	PGMD9
PD2	PGMD10
PD3	PGMD11
PD4	PGMD12
PD5	PGMD13
PD6	PGMD14
PD7	PGMD15

Downloaded from Arrow.com.

10.1.5.9 SAM-BA Boot

The SAM-BA Boot is a default boot program which provides an easy way to program in-situ the on-chip Flash memory.

The SAM-BA Boot Assistant supports serial communication via the UART0 and USB.

The SAM-BA Boot provides an interface with SAM-BA computer application.

The SAM-BA Boot is in ROM at address 0x0 when the bit GPNVM1 is set to 0.

10.1.5.10 General-purpose NVM (GPNVM) Bits

All SAM E70 devices feature nine general-purpose NVM (GPNVM) bits that can be cleared or set, respectively, through the “Clear GPNVM Bit” and “Set GPNVM Bit” commands of the EEFC User Interface.

The bit GPNVM0 is the security bit.

The bit GPNVM1 is used to select the Boot mode (Boot always at 0x00) on ROM or Flash.

Table 10-4. General-purpose Non volatile Memory Bits

GPNVM Bit	Function
0	Security bit
1	Boot mode selection 0: ROM (default) 1: Flash
5:2	Free
6	Reserved
8:7	TCM configuration 00: 0 Kbytes DTCM + 0 Kbytes ITCM (default) 01: 32 Kbytes DTCM + 32 Kbytes ITCM 10: 64 Kbytes DTCM + 64 Kbytes ITCM 11: 128 Kbytes DTCM + 128 Kbytes ITCM Note: After programming, a user reboot must be done.

10.1.6 Boot Strategies

The system always boots at address 0x0. To ensure maximum boot possibilities, the memory layout can be changed using GPNVM bits.

A GPNVM bit is used to boot either on the ROM (default) or from the Flash.

The GPNVM bit can be cleared or set, respectively, through the commands “Clear General-purpose NVM Bit” and “Set General-purpose NVM Bit” of the EEFC User Interface.

Setting the bit GPNVM1 selects boot from the Flash. Clearing it selects boot from the ROM. Asserting ERASE sets the bit GPNVM1 and thus selects boot from ROM.

10.2 External Memories

The SAM E70 features one External Bus Interface to provide an interface to a wide range of external memories and to any parallel peripheral.

Downloaded from Arrow.com.

11. Event System

The events generated by peripherals (source) are designed to be directly routed to peripherals (destination) using these events without processor intervention. The trigger source can be programmed in the destination peripheral.

11.1 Embedded Characteristics

- Timers, PWM, IOs and peripherals generate event triggers which are directly routed to destination peripherals such as AFEC or DACC to start measurement/conversion without processor intervention.
- UART, USART, QSPI, SPI, TWI, PWM, HSMCI, AES, AFEC, DACC, PIO, TC (Capture mode) also generate event triggers directly connected to the DMA Controller for data transfer without processor intervention.
- Parallel capture logic is directly embedded in the PIO and generates trigger events to the DMA Controller to capture data without processor intervention.
- PWM safety events (faults) are in combinational form and directly routed from event generators (AFEC, ACC, PMC, TC) to the PWM module.
- PWM output comparators (OCx) generate events directly connected to the TC.
- PMC safety event (clock failure detection) can be programmed to switch the MCK on reliable main RC internal clock without processor intervention.

Downloaded from Arrow.com.

11.2 Real-time Event Mapping

Table 11-1. Real-time Event Mapping List

Function	Application	Description	Event Source	Event Destination
Safety	General-purpose	Automatic switch to reliable main RC oscillator in case of main crystal clock failure[1]	Power Management Controller (PMC)	PMC
	General-purpose, motor control, power factor correction (PFC)	Puts the PWM outputs in Safe mode in case of main crystal clock failure[1][2]	PMC	Pulse Width Modulation 0 and 1 (PWM0 and PWM1)
	Motor control, PFC	Puts the PWM outputs in Safe mode (overcurrent detection, etc.)[2][3]	Analog Comparator Controller (ACC)	PWM0 and PWM1
	Motor control, PFC	Puts the PWM outputs in Safe mode (overspeed, overcurrent detection, etc.)[2][4]	Analog Front-End Controller (AFEC0)	PWM0 and PWM1
			AFEC1	PWM0 and PWM1
	Motor control	Puts the PWM outputs in Safe mode (overspeed detection through timer quadrature decoder)[2][6]	Timer Counter Block 0	PWM0
			Timer Counter Block 1	PWM1
	General-purpose, motor control, power factor correction (PFC)	Puts the PWM outputs in Safe mode (general-purpose fault inputs)[2]	PIO PA9, PD8, PD9	PWM0
			PIO PA21, PA26, PA28	PWM1
Security	General-purpose	Immediate GPBR clear (asynchronous) on tamper detection through WKUP0/1 IO pins [5]	PIO WKUP0/1	GPBR
Measurement trigger	Power factor correction (DC-DC, lighting, etc.)	Duty cycle output waveform correction Trigger source selection in PWM[7][8]	ACC	PWM0
			PIO PA10, PA22	PWM0
			ACC	PWM1
			PIO PA30, PA18	PWM1
	General-purpose	Trigger source selection in AFEC[9]	PIO AFE0_ADTRG	AFEC0
			TC0 TIOA0	AFEC0
			TC0 TIOA1	AFEC0
			TC0 TIOA2	AFEC0
			ACC	AFEC0
	Motor control	ADC-PWM synchronization[12][14] Trigger source selection in AFEC[9]	PWM0 Event Line 0 and 1	AFEC0
	General-purpose	Trigger source selection in AFEC[9]	PIO AFE1_ADTRG	AFEC1
			TC1 TIOA3	AFEC1
			TC1 TIOA4	AFEC1
			TC1 TIOA5	AFEC1
			ACC	AFEC1
	Motor control	ADC-PWM synchronization[12][14] Trigger source selection in AFEC[9]	PWM1 Event Line 0 and 1	AFEC1
	General-purpose	Temperature sensor Low-speed measurement [10][11]	RTC RTCOUT0	AFEC0 and AFEC1

Downloaded from Arrow.com.

Table 11-1. Real-time Event Mapping List (Continued)

Function	Application	Description	Event Source	Event Destination
Conversion trigger	General-purpose	Trigger source selection in DACC (Digital-to-Analog Converter Controller)[13]	TC0 TIOA0, TIOA1, TIOA2	DACC
			PIO DATRG	DACC
			PWM0 Event Line 0 and 1[14]	DACC
			PWM1 Event Line 0 and 1[14]	DACC
Image capture	Low-cost image sensor	Direct image transfer from sensor to system memory via DMA[15]	PIO PA3/4/5/9/10/11/12/13, PA22, PA14, PA21	DMA
Delay measurement	Motor control	Propagation delay of external components (IOs, power transistor bridge driver, etc.)[16][17]	PWM0 Comparator Output OC0	TC TIOA0 and TIOB0
			PWM0 Comparator Output OC1	TC TIOA1 and TIOB1
			PWM0 Comparator Output OC2	TC TIOA2 and TIOB2
			PWM1 Comparator Output OC0	TC TIOA3 and TIOB3
			PWM1 Comparator Output OC1	TC TIOA4 and TIOB4
			PWM1 Comparator Output OC2	TC TIOA5 and TIOB5
			PWM0 Comparator Output OC0	TC TIOA6 and TIOB6
			PWM0 Comparator Output OC1	TC TIOA7 and TIOB7
			PWM0 Comparator Output OC2	TC TIOA8 and TIOB8
			PWM1 Comparator Output OC0	TC TIOA9 and TIOB9
			PWM1 Comparator Output OC1	TC TIOA10 and TIOB10
Audio clock recovery from Ethernet	Audio	GMAC GTSUCOMP signal adaptation via TC (TC_EMR.TRIGSRCB) in order to drive the clock reference of the external PLL for the audio clock	GMAC GTSUCOMP	TC TIOB11
Direct Memory Access	General-purpose	Peripheral trigger event generation to transfer data to/from system memory[18]	USART, UART, TWIHS, SPI, QSPI, AFEC, TC (Capture), SSC, HSMCI, DAC, AES, PWM, PIO, I2SC	XDMA

Notes:
1. Refer to Section 31.15 “Main Clock Failure Detection”.
2. Refer to Section 49.5.4 “Fault Inputs” and Section 49.6.2.7 “Fault Protection”.
3. Refer to Section 52.6.4 “Fault Mode”.
4. Refer to Section 50.5.7 “Fault Output”.
5. Refer to Section 22.4.9.2 “Low-power Tamper Detection and Anti-Tampering” and Section 29.3.1 “General Purpose Backup Register x”.
6. Refer to Section 48.6.18 “Fault Mode”.
7. Refer to Section 49.7.49 “PWM External Trigger Register”.
8. Refer to Section 49.6.5 “PWM External Trigger Mode”.
9. Refer to Section 50.6.6 “Conversion Triggers” and Section 50.7.2 “AFEC Mode Register”.
10. Refer to Section 50.5.4 “Temperature Sensor”.

Downloaded from Arrow.com.

11. Refer to Section 26.5.8 “Waveform Generation”.
12. Refer to Section 49.7.36 “PWM Comparison x Value Register”.
13. Refer to Section 51.7.3 “DACC Trigger Register”.
14. Refer to Section 49.6.3 “PWM Comparison Units” and Section 49.6.4 “PWM Event Lines”.
15. Refer to Section 32.5.14 “Parallel Capture Mode”.
16. Refer to Section 49.6.2.2 “Comparator”.
17. Refer to Section 48.6.14 “Synchronization with PWM”.
18. Refer to Section 35. “DMA Controller (XDMAC)”.

Downloaded from Arrow.com.

12. System Controller

The System Controller is a set of peripherals that handles key elements of the system, such as power, resets, clocks, time, interrupts, watchdog, etc.

12.1 System Controller and Peripherals Mapping

Refer to Section 9. “Product Mapping”.

12.2 Power-on-Reset, Brownout and Supply Monitor

The SAM E70 embeds three features to monitor, warn and/or reset the chip:

- Power-on-Reset on VDDIO
- Power-on-Reset on VDDCORE
- Brownout Detector on VDDCORE
- Supply Monitor on VDDIO

12.2.1 Power-on-Reset

The Power-on-Reset monitors VDDIO and VDDCORE. It is always activated and monitors voltage at start up but also during power down. If VDDIO or VDDCORE goes below the threshold voltage, the entire chip is reset. For more information, refer to Section 56. “Electrical Characteristics”.

12.2.2 Brownout Detector on VDDCORE

The Brownout Detector monitors VDDCORE. It is active by default. It can be deactivated by software through the Supply Controller (SUPC_MR). It is especially recommended to disable it during low-power modes such as wait or sleep modes.

If VDDCORE goes below the threshold voltage, the reset of the core is asserted. For more information, refer to Section 22. “Supply Controller (SUPC)” and Section 56. “Electrical Characteristics”.

12.2.3 Supply Monitor on VDDIO

The Supply Monitor monitors VDDIO. It is not active by default. It can be activated by software and is fully programmable with 16 steps for the threshold (between 1.6V to 3.4V). It is controlled by the Supply Controller (SUPC). A sample mode is possible. It allows to divide the supply monitor power consumption by a factor of up to 2048. For more information, refer to Section 22. “Supply Controller (SUPC)” and Section 56. “Electrical Characteristics”.

Downloaded from Arrow.com.

12.3 Reset Controller

The Reset Controller is based on two Power-on-Reset cells, one on VDDIO and one on VDDCORE, and a Supply Monitor on VDDIO.

The Reset Controller returns the source of the last reset to the software. This may be a general reset, a wake-up reset, a software reset, a user reset or a watchdog reset.

The Reset Controller controls the internal resets of the system and the pin input/output. It can shape a reset signal for the external devices, simplifying the connection of a push-button on the NRST pin to implement a manual reset.

The configuration of the Reset Controller is saved as supplied on VDDIO.

Downloaded from Arrow.com.

13. Peripherals

13.1 Peripheral Identifiers

Table 13-1 defines the peripheral identifiers of the SAM E70. A peripheral identifier is required for the control of the peripheral interrupt with the Nested Vectored Interrupt Controller and control of the peripheral clock with the Power Management Controller.

Table 13-1. Peripheral Identifiers

Instance ID	Instance Name	NVIC Interrupt	PMC Clock Control	Description
0	SUPC	X	–	Supply Controller
1	RSTC	X	–	Reset Controller
2	RTC	X	–	Real Time Clock
3	RTT	X	–	Real Time Timer
4	WDT	X	–	Watchdog Timer
5	PMC	X	–	Power Management Controller
6	EFC	X	–	Enhanced Embedded Flash Controller
7	UART0	X	X	Universal Asynchronous Receiver/Transmitter
8	UART1	X	X	Universal Asynchronous Receiver/Transmitter
9	SMC	–	X	Static Memory Controller
10	PIOA	X	X	Parallel I/O Controller A
11	PIOB	X	X	Parallel I/O Controller B
12	PIOC	X	X	Parallel I/O Controller C
13	USART0	X	X	Universal Synchronous/Asynchronous Receiver/Transmitter
14	USART1	X	X	Universal Synchronous/Asynchronous Receiver/Transmitter
15	USART2	X	X	Universal Synchronous/Asynchronous Receiver/Transmitter
16	PIOD	X	X	Parallel I/O Controller D
17	PIOE	X	X	Parallel I/O Controller E
18	HSMCI	X	X	Multimedia Card Interface
19	TWIHS0	X	X	Two-wire Interface
20	TWIHS1	X	X	Two-wire Interface
21	SPI0	X	X	Serial Peripheral Interface
22	SSC	X	X	Synchronous Serial Controller
23	TC0	X	X	16-bit Timer Counter Channel 0
24	TC1	X	X	16-bit Timer Counter Channel 1
25	TC2	X	X	16-bit Timer Counter Channel 2
26	TC3	X	X	16-bit Timer Counter Channel 3
27	TC4	X	X	16-bit Timer Counter Channel 4
28	TC5	X	X	16-bit Timer Counter Channel 5

Downloaded from Arrow.com.

Table 13-1. Peripheral Identifiers (Continued)

Instance ID	Instance Name	NVIC Interrupt	PMC Clock Control	Description
29	AFEC0	X	X	Analog Front-End Controller
30	DACC	X	X	Digital-to-Analog Converter
31	PWM0	X	X	Pulse Width Modulation Controller
32	ICM	X	X	Integrity Check Monitor
33	ACC	X	X	Analog Comparator Controller
34	USBHS	X	X	USB Host / Device Controller
35	MCAN0	X	X	CAN IRQ Line 0
36	MCAN0	INT1	–	CAN IRQ Line 1
37	MCAN1	X	X	CAN IRQ Line 0
38	MCAN1	INT1	–	CAN IRQ Line 1
39	GMAC	X	X	Ethernet MAC
40	AFEC1	X	X	Analog Front End Controller
41	TWIHS2	X	X	Two-wire Interface
42	SPI1	X	X	Serial Peripheral Interface
43	QSPI	X	X	Quad I/O Serial Peripheral Interface
44	UART2	X	X	Universal Asynchronous Receiver/Transmitter
45	UART3	X	X	Universal Asynchronous Receiver/Transmitter
46	UART4	X	X	Universal Asynchronous Receiver/Transmitter
47	TC6	X	X	16-bit Timer Counter Channel 6
48	TC7	X	X	16-bit Timer Counter Channel 7
49	TC8	X	X	16-bit Timer Counter Channel 8
50	TC9	X	X	16-bit Timer Counter Channel 9
51	TC10	X	X	16-bit Timer Counter Channel 10
52	TC11	X	X	16-bit Timer Counter Channel 11
53	–	–	–	Reserved
54	–	–	–	Reserved
55	–	–	–	Reserved
56	AES	X	X	Advanced Encryption Standard
57	TRNG	X	X	True Random Number Generator
58	XDMAC	X	X	DMA Controller
59	ISI	X	X	Image Sensor Interface
60	PWM1	X	X	Pulse Width Modulation Controller
61	ARM	FPU	–	ARM Floating Point Unit interrupt associated with OFC, UFC, IOC, DZC and IDC bits
62	SDRAMC	X	–	SDRAM Controller
63	RSWDT	X	–	Reinforced Safety Watchdog Timer
64	ARM	CCW	–	ARM Cache ECC Warning

Downloaded from Arrow.com.

Table 13-1. Peripheral Identifiers (Continued)

Instance ID	Instance Name	NVIC Interrupt	PMC Clock Control	Description
65	ARM	CCF	–	ARM Cache ECC Fault
66	GMAC	Q1	–	GMAC Queue 1 Interrupt signal toggled on a DMA write to the first word of each DMA data buffer associated with queue 1
67	GMAC	Q2	–	GMAC Queue 2 Interrupt signal toggled on a DMA write to the first word of each DMA data buffer associated with queue 2
68	ARM	IXC	–	Floating Point Unit Interrupt IXC associated with FPU cumulative exception bit
69	I2SC0	X	X	Inter-IC Sound Controller
70	I2SC1	X	X	Inter-IC Sound Controller

Downloaded from Arrow.com.

13.2 Peripheral Signal Multiplexing on I/O Lines

The SAM E70 features

- two PIO controllers on 64-pin versions (PIOA and PIOB)
- three PIO controllers on the 100-pin version (PIOA, PIOB and PIOD)
- five PIO controllers on the 144-pin version (PIOA, PIOB, PIOC, PIOD and PIOE), that multiplex the I/O lines of the peripheral set.

The SAM E70 PIO Controllers control up to 32 lines. Each line can be assigned to one of four peripheral functions: A, B, C or D.

For more information on multiplexed signals, refer to Section 5. “Package and Pinout”.

Downloaded from Arrow.com.

14. ARM Cortex-M7 Processor

14.1 Description

The ARM Cortex-M7 processor implements the ARMv7-M architecture and runs 32-bit ARM instructions, 16-bit and 32-bit Thumb instructions.

The double-precision Floating-Point Unit (FPU) supports the ARMv7 VFPv5 architecture. It is tightly integrated to the ARM Cortex-M7 processor pipeline. It provides trapless execution and is optimized for scalar operation. It can generate an Undefined instruction exception on vector instructions that enables the programmer to emulate vector capability in software.

Note: Refer to ARM reference documents *Cortex-M7 Processor User Guide* (ARM DUI 0644) and *Cortex-M7 Technical Reference Manual* (A*RM DDI 0489)*, available on www.arm.com.

14.1.1 System-Level Interface

The ARM Cortex-M7 processor provides multiple interfaces using AMBA technology to provide high-speed, low-latency memory accesses. It supports unaligned data accesses and implements atomic bit manipulation that enables faster peripheral controls, system spinlocks and thread-safe Boolean data handling.

The ARM Cortex-M7 processor has a Memory Protection Unit (MPU) that provides fine-grain memory control, enabling applications to utilize multiple privilege levels, separating and protecting code, data and stack on a task-by-task basis. Such requirements are becoming critical in many embedded applications such as automotive.

14.1.2 Integrated Configurable Debug

The ARM Cortex-M7 processor implements a complete hardware debug solution. This provides high system visibility of the processor and memory through a 2-pin Serial Wire Debug (SWD) port that is ideal for microcontrollers and other small package devices.

For system trace, the processor integrates an Instrumentation Trace Macrocell (ITM) alongside data watchpoints and a profiling unit. To enable simple and cost-effective profiling of the system events these generate, a Serial Wire Viewer (SWV) can export a stream of software-generated messages, data trace, and profiling information through a single pin.

The Embedded Trace Macrocell (ETM) delivers unrivalled instruction trace capture in an area far smaller than traditional trace units, enabling many low-cost MCUs to implement full instruction trace for the first time.

The Flash Patch and Breakpoint Unit (FPB) provides up to eight hardware breakpoint comparators that debuggers can use. The comparators in the FPB also provide remap functions of up to eight words in the program code in the CODE memory region. This enables applications stored on a non-erasable, ROM-based microcontroller to be patched if a small programmable memory, for example Flash, is available in the device. During initialization, the application in ROM detects, from the programmable memory, whether a patch is required. If a patch is required, the application programs the FPB to remap a number of addresses. When those addresses are accessed, the accesses are redirected to a remap table specified in the FPB configuration, which means the program in the non-modifiable ROM can be patched.

Downloaded from Arrow.com.

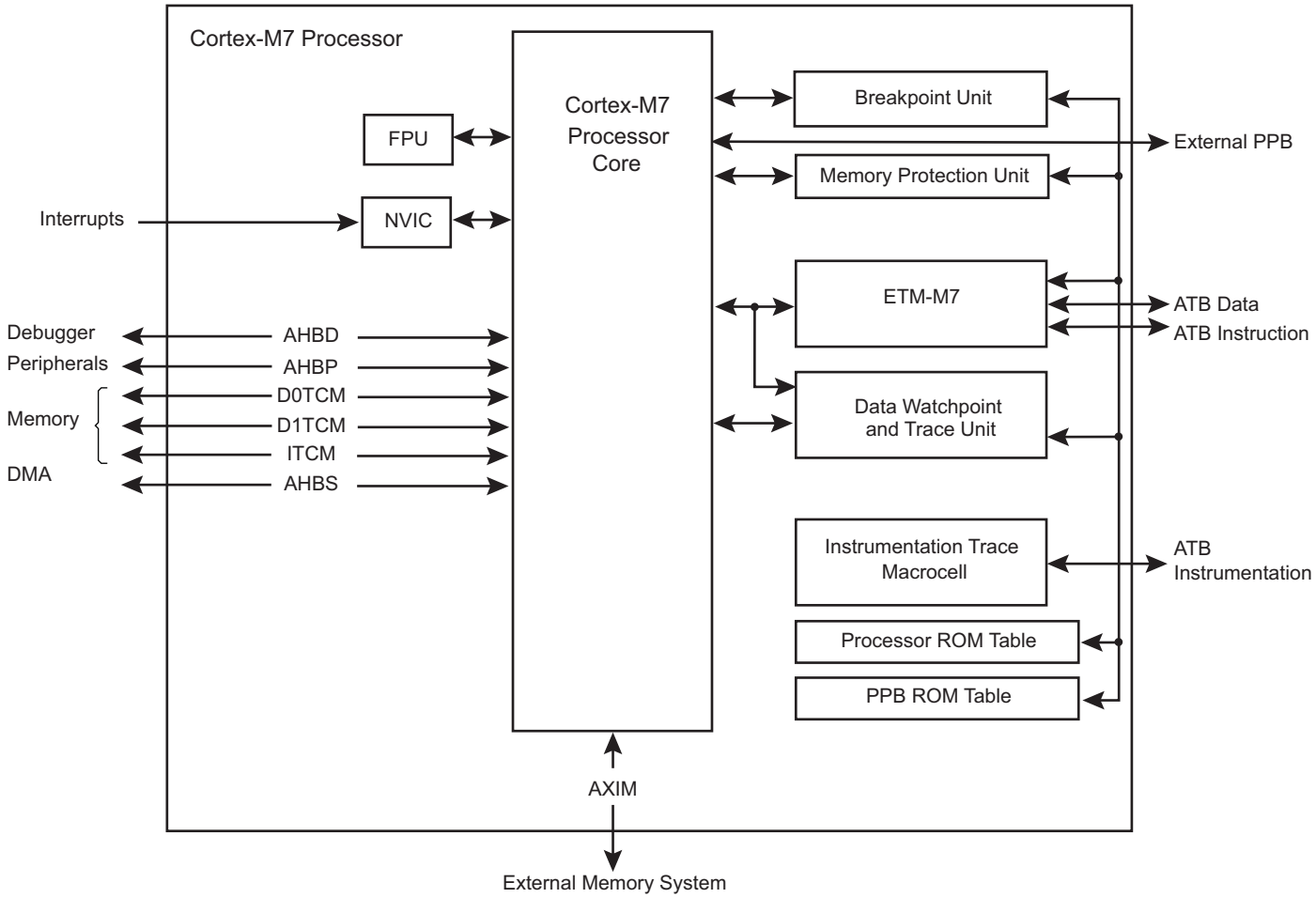
14.2 Embedded Characteristics

- ARM Cortex-M7 with 16 KB of instruction cache and 16 KB of data cache
- ARMv7-M Thumb instruction set combines high-code density with 32-bit performance
- Tightly Coupled Memory (TCM) interfaces:
  - 64-bit ITCM interface
  - 2 x 32-bit DTCM interfaces
- Memory Protection Unit (MPU): up to 16 protected memory regions for safety/critical applications
- Dedicated low-latency AHB-Lite peripheral (AHBP) interface
- Dedicated AHB slave (AHBS) interface for system access to TCMs
- Low-latency interrupt processing achieved by a Nested Vectored Interrupt Controller (NVIC) closely integrated with the processor
- DSP extensions for efficient signal processing and complex algorithm execution
- IEEE Standard 754-2008 Floating Point Unit (FPU)
- Hardware integer divide instructions
- Extensive debug and trace capabilities:
  - Serial Wire Debug and Serial Wire Trace reduce the number of pins required for debugging, tracing, and code profiling

Downloaded from Arrow.com.

14.3 Block Diagram

Figure 14-1. ARM Cortex-M7 Processor Top-level Diagram



Downloaded from Arrow.com.

14.4 Programmer's Model

This section describes the ARM Cortex-M7 programmer's model. In addition to the individual core register descriptions, it contains information about the processor modes and privilege levels for software execution and stacks.

14.4.1 Processor Modes and Privilege Levels for Software Execution

The processor modes are:

- Thread mode
  Used to execute application software. The processor enters Thread mode when it comes out of reset.
- Handler mode
  Used to handle exceptions. The processor returns to Thread mode when it has finished exception processing.

The privilege levels for software execution are:

- Unprivileged
  The software:
  - Has limited access to the MSR and MRS instructions, and cannot use the CPS instruction
  - Cannot access the System Timer, NVIC, or System Control Block
  - Might have a restricted access to memory or peripherals.

Unprivileged software executes at the unprivileged level.

- Privileged
  The software can use all the instructions and has access to all resources. Privileged software executes at the privileged level.

In Thread mode, the Control Register controls whether the software execution is privileged or unprivileged, see "Control Register" . In Handler mode, software execution is always privileged.

Only privileged software can write to the Control Register to change the privilege level for software execution in Thread mode. Unprivileged software can use the SVC instruction to make a supervisor call to transfer control to privileged software.

14.4.2 Stacks

The processor uses a full descending stack. This means the stack pointer holds the address of the last stacked item in memory When the processor pushes a new item onto the stack, it decrements the stack pointer and then writes the item to the new memory location. The processor implements two stacks, the main stack and the process stack, with a pointer for each held in independent registers, see "Stack Pointer" .

In Thread mode, the Control Register controls whether the processor uses the main stack or the process stack, see "Control Register" .

In Handler mode, the processor always uses the main stack.

The options for processor operations are:

Table 14-1. Summary of Processor Mode, Execution Privilege Level, and Stack Use Options

Processor Mode	Used to Execute	Privilege Level for Software Execution	Stack Used
Thread	Applications	Privileged or unprivileged(1)	Main stack or process stack(1)
Handler	Exception handlers	Always privileged	Main stack

Note: 1. See "Control Register" .

Downloaded from Arrow.com.

14.4.2.1 Processor Core Registers

Figure 14-2. Processor Core Registers

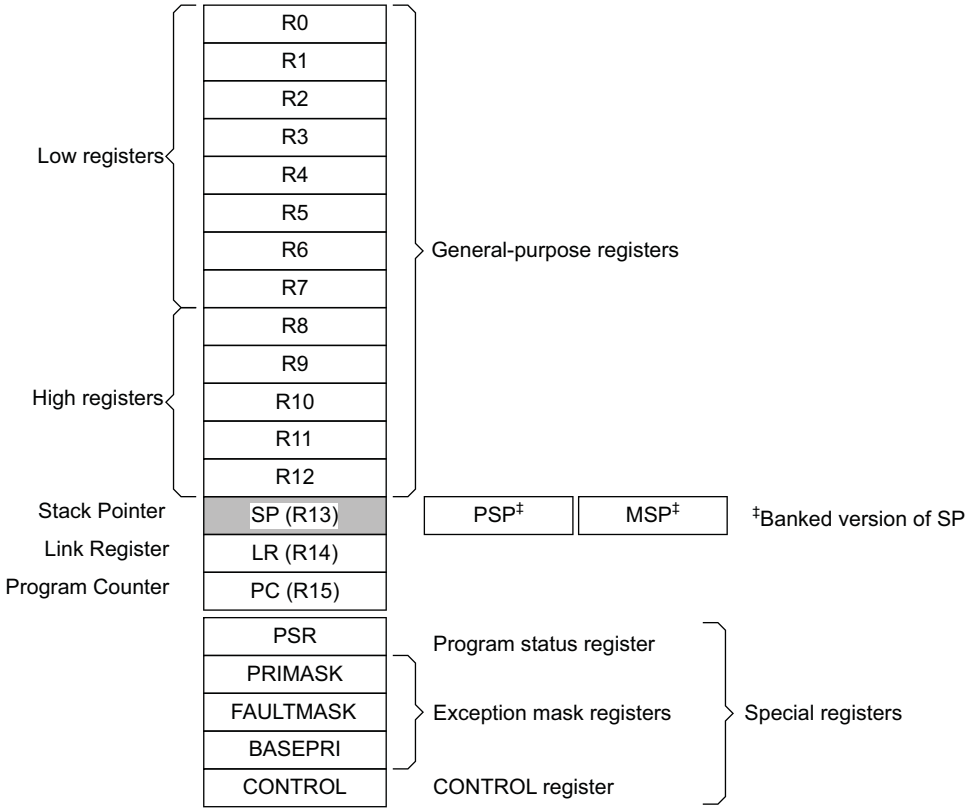


Table 14-2. Processor Core Registers

Register	Name	Access[1]	Required Privilege[2]	Reset
General-purpose registers	R0–R12	Read/Write	Either	Unknown
Stack Pointer	MSP	Read/Write	Privileged	See Section 14.4.4
Stack Pointer	PSP	Read/Write	Either	Unknown
Link Register	LR	Read/Write	Either	0xFFFFFFFF
Program Counter	PC	Read/Write	Either	See Section 14.4.6.1
Program Status Register	PSR	Read/Write	Privileged	0x01000000
Application Program Status Register	APSR	Read/Write	Either	0x00000000
Interrupt Program Status Register	IPSR	Read-only	Privileged	0x00000000
Execution Program Status Register	EPSR	Read-only	Privileged	0x01000000
Priority Mask Register	PRIMASK	Read/Write	Privileged	0x00000000
Fault Mask Register	FAULTMASK	Read/Write	Privileged	0x00000000
Base Priority Mask Register	BASEPRI	Read/Write	Privileged	0x00000000
Control Register	CONTROL	Read/Write	Privileged	0x00000000

Notes: 1. Describes access type during program execution in Thread mode and Handler mode. Debug access can differ.
2. An entry of Either means privileged and unprivileged software can access the register.

Downloaded from Arrow.com.

14.4.3 General-purpose Registers

R0–R12 are 32-bit general-purpose registers for data operations.

14.4.4 Stack Pointer

The Stack Pointer (SP) is register R13. In Thread mode, bit[1] of the Control Register indicates the stack pointer to use:

- 0 = Main Stack Pointer (MSP). This is the reset value.
- 1 = Process Stack Pointer (PSP).

On reset, the processor loads the MSP with the value from address 0x00000000.

14.4.5 Link Register

The Link Register (LR) is register R14. It stores the return information for subroutines, function calls, and exceptions. On reset, the processor loads the LR value 0xFFFFFFFF.

14.4.6 Program Counter

The Program Counter (PC) is register R15. It contains the current program address. On reset, the processor loads the PC with the value of the reset vector, which is at address 0x00000004. Bit[0] of the value is loaded into the EPSR T-bit at reset and must be 1.

Downloaded from Arrow.com.

14.4.6.1 Program Status Register

Name: PSR

Access: Read/Write

31	30	29	28	27	26	25	24
N	Z	C	V	Q	ICI/IT		T

23	22	21	20	19	18	17	16
–							

15	14	13	12	11	10	9	8
ICI/IT						–	ISR_NUMBER

7	6	5	4	3	2	1	0
ISR_NUMBER							

The Program Status Register (PSR) combines:

- Application Program Status Register (APSR)
- Interrupt Program Status Register (IPSR)
- Execution Program Status Register (EPSR).

These registers are mutually exclusive bitfields in the 32-bit PSR.

The PSR accesses these registers individually or as a combination of any two or all three registers, using the register name as an argument to the MSR or MRS instructions. For example:

- Read of all the registers using PSR with the MRS instruction
- Write to the APSR N, Z, C, V and Q bits using APSR_nzcvq with the MSR instruction.

The PSR combinations and attributes are:

Name	Access	Combination
PSR	Read/Write(1)(2)	APSR, EPSR, and IPSR
IEPSR	Read-only	EPSR and IPSR
IAPSR	Read/Write(1)	APSR and IPSR
EAPSR	Read/Write(2)	APSR and EPSR

Notes: 1. The processor ignores writes to the IPSR bits.
2. Reads of the EPSR bits return zero, and the processor ignores writes to these bits.

For more information about how to access the program status registers, see description of instructions “MRS” and “MSR” in the relevant ARM documentation.

Downloaded from Arrow.com.

14.4.6.2 Application Program Status Register

Name: APSR

Access: Read/Write

31	30	29	28	27	26	25	24
N	Z	C	V	Q	–		

23	22	21	20	19	18	17	16
–				GE[3:0]			

15	14	13	12	11	10	9	8
–							

7	6	5	4	3	2	1	0
–							

The APSR contains the current state of the condition flags from previous instruction executions.

- **N: Negative Flag**

0: Operation result was positive, zero, greater than, or equal

1: Operation result was negative or less than.

- **Z: Zero Flag**

0: Operation result was not zero

1: Operation result was zero.

- **C: Carry or Borrow Flag**

Carry or borrow flag:

0: Add operation did not result in a carry bit or subtract operation resulted in a borrow bit

1: Add operation resulted in a carry bit or subtract operation did not result in a borrow bit.

- **V: Overflow Flag**

0: Operation did not result in an overflow

1: Operation resulted in an overflow.

- **Q: DSP Overflow and Saturation Flag**

Sticky saturation flag:

0: Indicates that saturation has not occurred since reset or since the bit was last cleared to zero

1: Indicates when an SSAT or USAT instruction results in saturation.

This bit is cleared to zero by software using an MRS instruction.

- **GE[19:16]: Greater Than or Equal Flags**

For more information, see description of the “SEL” instruction in the relevant ARM documentation.

Downloaded from Arrow.com.

14.4.6.3 Interrupt Program Status Register

Name: IPSR

Access: Read/Write

31	30	29	28	27	26	25	24
–							

23	22	21	20	19	18	17	16
–							

15	14	13	12	11	10	9	8
–							ISR_NUMBER

7	6	5	4	3	2	1	0
ISR_NUMBER							

The IPSR contains the exception type number of the current Interrupt Service Routine (ISR).

- **ISR_NUMBER: Number of the Current Exception**

0 = Thread mode

1 = Reserved

2 = NMI

3 = Hard fault

4 = Memory management fault

5 = Bus fault

6 = Usage fault

7–10 = Reserved

11 = SVCall

12 = Reserved for Debug

13 = Reserved

14 = PendSV

15 = SysTick

16 = IRQ0

...

75 = IRQ72

For more information, see “Exception Types” in the relevant ARM documentation.

Downloaded from Arrow.com.

14.4.6.4 Execution Program Status Register

Name: EPSR

Access: Read/Write

31	30	29	28	27	26	25	24
–					ICI/IT		T

23	22	21	20	19	18	17	16
–							

15	14	13	12	11	10	9	8
ICI/IT						–	

7	6	5	4	3	2	1	0
–							

The EPSR contains the Thumb state bit, and the execution state bits for either the If-Then (IT) instruction, or the Interruptible-Continuable Instruction (ICI) field for an interrupted load multiple or store multiple instruction.

Attempts to read the EPSR directly through application software using the MSR instruction always return zero. Attempts to write the EPSR using the MSR instruction in the application software are ignored. Fault handlers can examine the EPSR value in the stacked PSR to indicate the operation that is at fault. For more information, see “Exception Entry and Return” in the relevant ARM documentation.

- **ICI: Interruptible-continuable Instruction**

When an interrupt occurs during the execution of an LDM, STM, PUSH, POP, VLDM, VSTM, VPUSH, or VPOP instruction, the processor:

– Stops the load multiple or store multiple instruction operation temporarily

– Stores the next register operand in the multiple operation to EPSR bits[15:12].

After servicing the interrupt, the processor:

– Returns to the register pointed to by bits[15:12]

– Resumes the execution of the multiple load or store instruction.

When the EPSR holds the ICI execution state, bits[26:25,11:10] are zero.

- **IT: If-Then Instruction**

Indicates the execution state bits of the IT instruction.

The If-Then block contains up to four instructions following an IT instruction. Each instruction in the block is conditional. The conditions for the instructions are either all the same, or some can be the inverse of others. For more information, see description of the “IT” instruction in the relevant ARM documentation.

- **T: Thumb State**

The ARM Cortex-M7 processor only supports the execution of instructions in Thumb state. The following can clear the T bit to 0:

– Instructions BLX, BX and POP{PC}

– Restoration from the stacked xPSR value on an exception return

– Bit[0] of the vector value on an exception entry or reset.

Attempting to execute instructions when the T bit is 0 results in a fault or lockup. For more information, see description of the “Lockup” instruction in the relevant ARM documentation.

Downloaded from Arrow.com.

14.4.6.5 Exception Mask Registers

The exception mask registers disable the handling of exceptions by the processor. Disable exceptions where they might impact on timing critical tasks.

To access the exception mask registers use the MSR and MRS instructions, or the CPS instruction to change the value of PRIMASK or FAULTMASK. For more information, see descriptions of the “MRS”, “MSR” and “CPS” instructions in the relevant ARM documentation.

14.4.6.6 Priority Mask Register

Name: PRIMASK

Access: Read/Write

31	30	29	28	27	26	25	24
–							

23	22	21	20	19	18	17	16
–							

15	14	13	12	11	10	9	8
–							

7	6	5	4	3	2	1	0
–							PRIMASK

The PRIMASK register prevents the activation of all exceptions with a configurable priority.

- **PRIMASK**

0: No effect.

1: Prevents the activation of all exceptions with a configurable priority.

Downloaded from Arrow.com.

14.4.6.7 Fault Mask Register

Name: FAULTMASK

Access: Read/Write

31	30	29	28	27	26	25	24
–							

23	22	21	20	19	18	17	16
–							

15	14	13	12	11	10	9	8
–							

7	6	5	4	3	2	1	0
–							FAULTMASK

The FAULTMASK register prevents the activation of all exceptions except for Non-Maskable Interrupt (NMI).

Downloaded from Arrow.com.

- **FAULTMASK**

0: No effect.

1: Prevents the activation of all exceptions except for NMI.

The processor clears the FAULTMASK bit to 0 on exit from any exception handler except the NMI handler.

14.4.6.8 Base Priority Mask Register

Name: BASEPRI

Access: Read/Write

31	30	29	28	27	26	25	24
–							

23	22	21	20	19	18	17	16
–							

15	14	13	12	11	10	9	8
–							

7	6	5	4	3	2	1	0
BASEPRI							

The BASEPRI register defines the minimum priority for exception processing. When BASEPRI is set to a nonzero value, it prevents the activation of all exceptions with same or lower priority level as the BASEPRI value.

- **BASEPRI**

Priority mask bits:

0: No effect

Nonzero: Defines the base priority for exception processing

The processor does not process any exception with a priority value greater than or equal to BASEPRI.

This field is similar to the priority fields in the interrupt priority registers. The processor implements only bits[7:4] of this field, bits[3:0] read as zero and ignore writes. See “Interrupt Priority Registers” in the relevant ARM documentation. Remember that higher priority field values correspond to lower exception priorities.

Downloaded from Arrow.com.

14.4.6.9 Control Register

Name: CONTROL

Access: Read/Write

31	30	29	28	27	26	25	24
–							

23	22	21	20	19	18	17	16
–							

15	14	13	12	11	10	9	8
–							

7	6	5	4	3	2	1	0
–					FPCA	SPSEL	nPRIV

The Control Register controls the stack used and the privilege level for software execution when the processor is in Thread mode and indicates whether the FPU state is active.

- **FPCA: Floating-point Context Active**

Indicates whether the floating-point context is currently active:

0: No floating-point context active.

1: Floating-point context active.

The ARM Cortex-M7 uses this bit to determine whether to preserve the floating-point state when processing an exception.

- **SPSEL: Active Stack Pointer**

Defines the current stack:

0: MSP is the current stack pointer.

1: PSP is the current stack pointer.

In Handler mode, this bit reads as zero and ignores writes. The ARM Cortex-M7 updates this bit automatically on exception return.

- **nPRIV: Thread Mode Privilege Level**

Defines the Thread mode privilege level:

0: Privileged.

1: Unprivileged.

Handler mode always uses the MSP, so the processor ignores explicit writes to the active stack pointer bit of the Control Register when in Handler mode. The exception entry and return mechanisms update the Control Register based on the EXC_RETURN value.

In an OS environment, ARM recommends that threads running in Thread mode use the process stack, and the kernel and exception handlers use the main stack.

By default, the Thread mode uses the MSP. To switch the stack pointer used in Thread mode to the PSP, either:

- Use the MSR instruction to set the Active stack pointer bit to 1, or
- Perform an exception return to Thread mode with the appropriate EXC_RETURN value.

Note: When changing the stack pointer, the software must use an ISB instruction immediately after the MSR instruction. This ensures that instructions after the ISB execute using the new stack pointer. For more information, see description of the “ISB” instruction in the relevant ARM documentation.

Downloaded from Arrow.com.

14.4.6.10 Exceptions and Interrupts

The ARM Cortex-M7 processor supports interrupts and system exceptions. The processor and the Nested Vectored Interrupt Controller (NVIC) prioritize and handle all exceptions. An exception changes the normal flow of software control. The processor uses the Handler mode to handle all exceptions except for reset. See “Exception Entry” and “Exception Return” in the relevant ARM documentation for more information.

The NVIC registers control interrupt handling. See “Nested Vectored Interrupt Controller (NVIC)” in the relevant ARM documentation for more information.

14.4.6.11 Data Types

The processor supports the following data types:

- 32-bit words
- 16-bit halfwords
- 8-bit bytes
- 32-bit single-precision floating point numbers
- 64-bit double-precision floating point numbers
- The processor manages all data memory accesses as little-endian. Instruction memory and Private Peripheral Bus (PPB) accesses are always little-endian. See “Memory Regions, Types and Attributes” in the relevant ARM documentation for more information.

14.4.6.12 Cortex Microcontroller Software Interface Standard (CMSIS)

For an ARM Cortex-M7 microcontroller system, the *Cortex Microcontroller Software Interface Standard* (CMSIS) defines:

- A common way to:
  - Access peripheral registers
  - Define exception vectors
- The names of:
  - The registers of the core peripherals
  - The core exception vectors
- A device-independent interface for RTOS kernels, including a debug channel.

The CMSIS includes address definitions and data structures for the core peripherals in the ARM Cortex-M7 processor.

The CMSIS simplifies the software development by enabling the reuse of template code and the combination of CMSIS-compliant software components from various middleware vendors. Software vendors can expand the CMSIS to include their peripheral definitions and access functions for those peripherals.

This document includes the register names defined by the CMSIS, and gives short descriptions of the CMSIS functions that address the processor core and the core peripherals.

Note: This document uses the register short names defined by the CMSIS. In a few cases, these differ from the architectural short names that might be used in other documents.

More information about the CMSIS can be found in the relevant ARM documentation.

Downloaded from Arrow.com.

14.5 ARM Cortex-M7 Configuration

Table 14-3 gives the configuration for the ARM Cortex-M7 processor.

Table 14-3. ARM Cortex-M7 Configuration

Features	Configuration
Debug	
Comparator set	Full comparator set: 4 DWT and 8 FPB comparators
ETM support	Instruction ETM interface
Internal Trace support (ITM)	ITM and DWT trace functionality implemented
CTI and WIC	Not embedded
TCM	
ITCM max size	128 KB
DTCM max size	128 KB
Cache	
Cache size	16 KB for instruction cache, 16 KB for data cache
Number of sets	256 for instruction cache, 128 for data cache
Number of ways	2 for instruction cache, 4 for data cache
Number of words per cache line	8 words (32 bytes)
ECC on Cache	Embedded
NVIC	
IRQ number	72
IRQ priority levels	8
MPU	
Number of regions	16
FPU	
FPU precision	Single and double precision
AHB Port	
AHBP addressing size	512 MB

For more details, refer to the ARM documentation referenced in Section 14.1

Downloaded from Arrow.com.

15. Debug and Test Features

15.1 Description

The device features a number of complementary debug and test capabilities. The Serial Wire Debug Port (SW-DP) is used for standard debugging functions, such as downloading code and single-stepping through programs. It also embeds a serial wire trace.

15.2 Embedded Characteristics

- Debug access to all memory and registers in the system, including Cortex-M register bank, when the core is running, halted, or held in reset.
- Serial Wire Debug Port (SW-DP) debug access
- Flash Patch and Breakpoint (FPB) unit for implementing breakpoints and code patches
- Data Watchpoint and Trace (DWT) unit for implementing watchpoints, data tracing, and system profiling
- Instrumentation Trace Macrocell (ITM) for support of printf style debugging
- 6-pin Embedded Trace Macrocell (ETM) for instruction trace stream, including CoreSight™ Trace Port Interface Unit (TPIU)
- IEEE1149.1 JTAG Boundary scan on All Digital Pins

15.3 Associated Documents

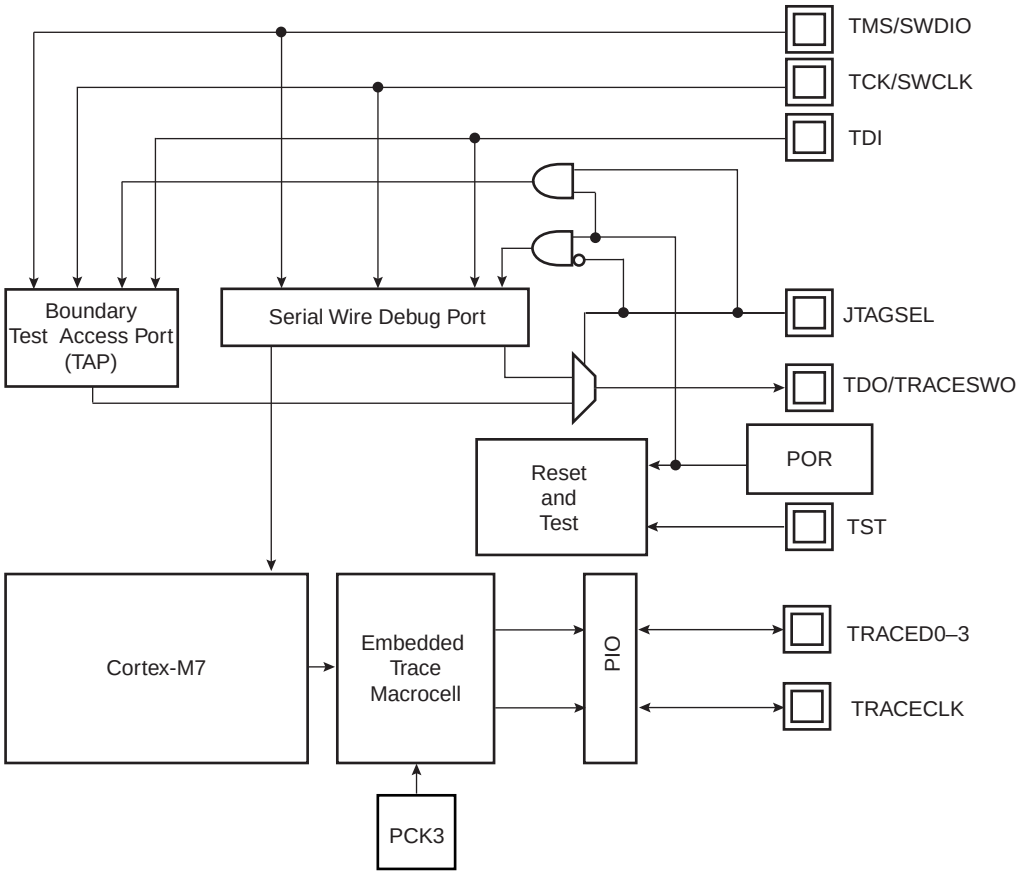
The SAM E70 implements the standard ARM CoreSight macrocell. For information on CoreSight, the following reference documents are available from the ARM web site:

- *Cortex-M7 User Guide Reference Manual* (ARM DUI 0644)
- *Cortex-M7 Technical Reference Manual* (ARM DDI 0489)
- *CoreSight Technology System Design Guide* (ARM DGI 0012)
- *CoreSight Components Technical Reference Manual* (ARM DDI 0314)
- *ARM Debug Interface v5 Architecture Specification* (Doc. ARM IHI 0031)
- *ARMv7-M Architecture Reference Manual* (ARM DDI 0403)

Downloaded from Arrow.com.

15.4 Debug and Test Block Diagram

Figure 15-1. Debug and Test Block Diagram



15.5 Debug and Test Pin Description

Table 15-1. Debug and Test Signal List

Signal Name	Function	Type	Active Level
Reset/Test			
NRST	Microcontroller Reset	Input/Output	Low
TST	Test Select	Input	–
Serial Wire Debug Port/JTAG Boundary Scan			
TCK/SWCLK	Test Clock/Serial Wire Clock	Input	–
TDI	Test Data In	Input	–
TDO/TRACESWO	Test Data Out/Trace Asynchronous Data Out	Output	–
TMS/SWDIO	Test Mode Select/Serial Wire Input/Output	Input	–
JTAGSEL	JTAG Selection	Input	High
Trace Debug Port			
TRACECLK	Trace Clock	Output	–
TRACED0–3	Trace Data	Output	–

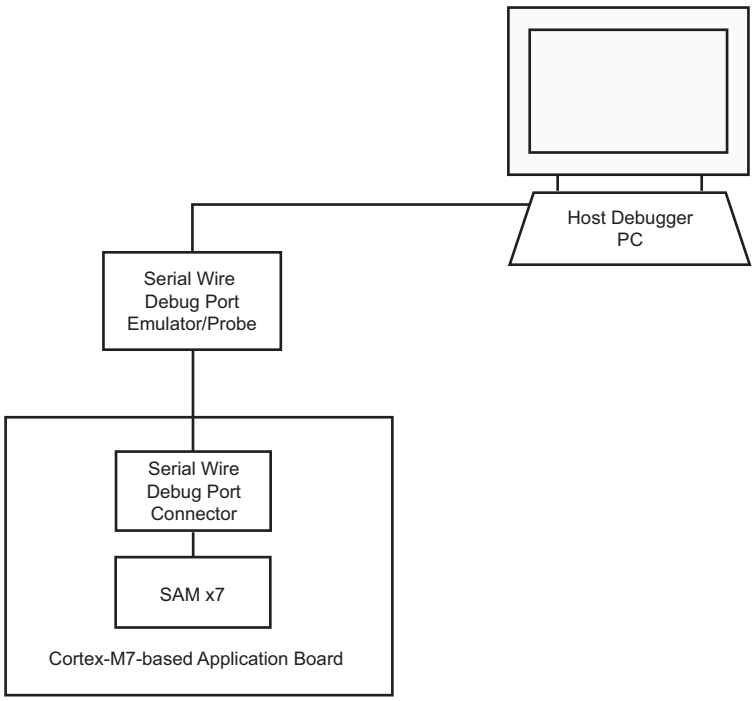
Downloaded from Arrow.com.

15.6 Application Examples

15.6.1 Debug Environment

Figure 15-2 shows a complete debug environment example. The SW-DP interface is used for standard debugging functions, such as downloading code and single-stepping through the program and viewing core and peripheral registers.

Figure 15-2. Application Debug Environment Example

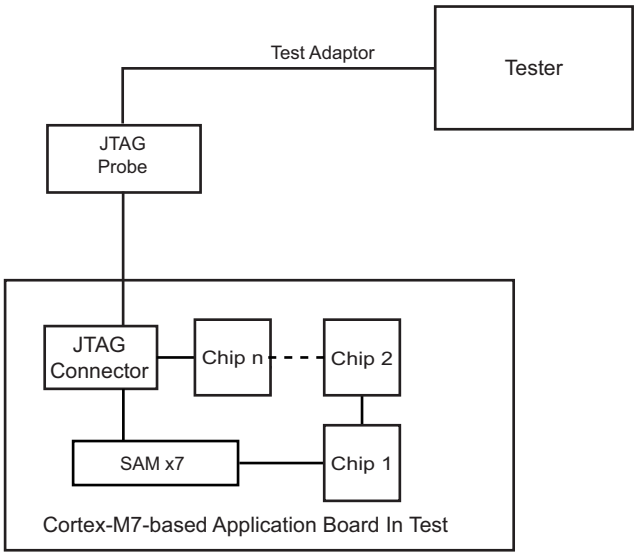


Downloaded from Arrow.com.

15.6.2 Test Environment

Figure 15-3 shows a test environment example (JTAG Boundary scan). Test vectors are sent and interpreted by the tester. In this example, the “board in test” is designed using a number of JTAG-compliant devices. These devices can be connected to form a single scan chain.

Figure 15-3. Application Test Environment Example



Downloaded from Arrow.com.

15.7 Functional Description

15.7.1 Test Pin

The TST pin is used for JTAG Boundary Scan Manufacturing Test or Fast Flash Programming mode. The TST pin integrates a permanent pull-down resistor of about 15 kΩ to GND, so that it can be left unconnected for normal operations. To enable Fast Flash Programming mode, refer to Section 17. “Fast Flash Programming Interface (FFPI)”.

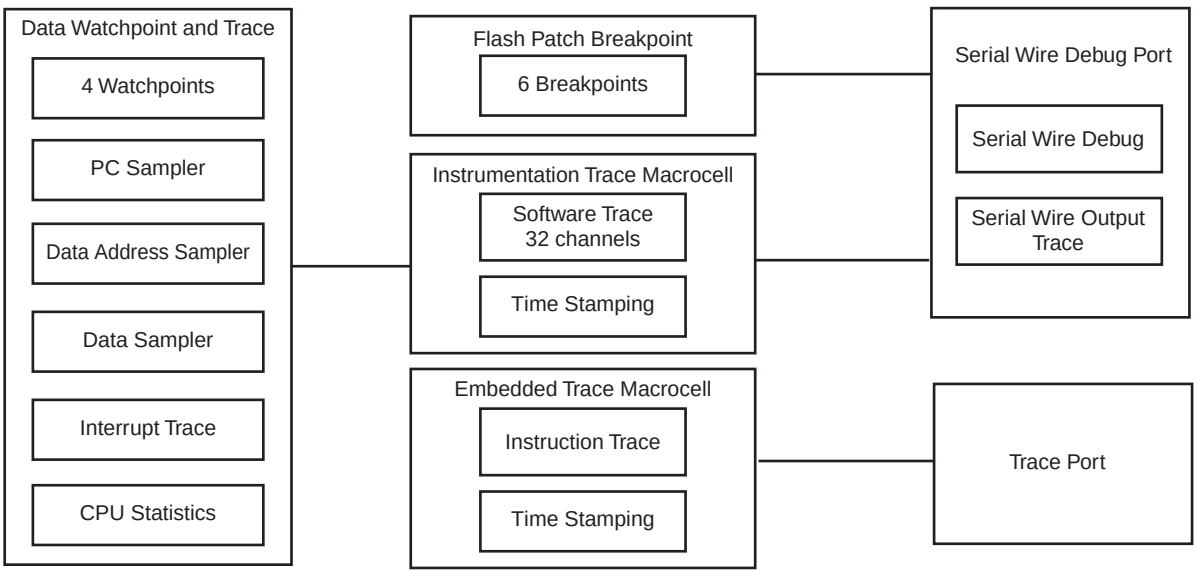
15.7.2 Debug Architecture

Figure 15-4 shows the debug architecture used. The Cortex-M7 embeds six functional units for debug:

- Serial Wire Debug Port (SW-DP) debug access
- FPB (Flash Patch Breakpoint)
- DWT (Data Watchpoint and Trace)
- ITM (Instrumentation Trace Macrocell)
- 6-pin Embedded Trace Macrocell (ETM) for instruction trace stream, including CoreSight Trace Port Interface Unit (TPIU)
- IEEE1149.1 JTAG Boundary scan on all digital pins

The debug architecture information that follows is mainly dedicated to developers of SW-DP Emulators/Probes and debugging tool vendors for Cortex-M7-based microcontrollers. For further details on SW-DP, see the *Cortex - M7 Technical Reference Manual*.

Figure 15-4. Debug Architecture



15.7.3 Serial Wire Debug Port (SW-DP) Pins

The SW-DP pins SWCLK and SWDIO are commonly provided on a standard 20-pin JTAG connector defined by ARM. For more details on voltage reference and reset state, refer to Section 4. “Signal Description”.

At startup, SW-DP pins are configured in SW-DP mode to allow connection with debugging probe.

SW-DP pins can be used as standard I/Os to provide users more general input/output pins when the debug port is not needed in the end application. Mode selection between SW-DP mode (System I/O mode) and general I/O mode is performed through the AHB Matrix Chip Configuration registers (CCFG_SYSIO). Configuration of the pad for pull-up, triggers, debouncing and glitch filters is possible regardless of the mode.

Downloaded from Arrow.com.

The JTAGSEL pin is used to select the JTAG boundary scan when asserted at a high level. It integrates a permanent pull-down resistor of about 15 kΩ to GND, so that it can be left unconnected for normal operations.

The JTAG debug ports TDI, TDO, TMS and TCK are inactive. They are provided for Boundary Scan Manufacturing Test purposes only. By default the SW-DP is active; TDO/TRACESWO can be used for trace.

Table 15-2. SW-DP Pin List

Pin Name	JTAG Boundary Scan	Serial Wire Debug Port
TMS/SWDIO	TMS	SWDIO
TCK/SWCLK	TCK	SWCLK
TDI	TDI	–
TDO/TRACESWO	TDO	TRACESWO (optional: trace)

SW-DP is selected when JTAGSEL is low. It is not possible to switch directly between SW-DP and JTAG boundary scan operations. A chip reset must be performed after JTAGSEL is changed.

Downloaded from Arrow.com.

15.7.4 Embedded Trace Module (ETM) Pins

The Embedded Trace Module (ETM) uses the Trace Port Interface Unit (TPIU) to export data out of the system.

The TPUI features the pins:

- TRACECLK–always exported to enable synchronization back with the data. PCK3 is used internally.
- TRACED0–3–the instruction trace stream.

15.7.5 Flash Patch Breakpoint (FPB)

The FPB implements hardware breakpoints.

15.7.6 Data Watchpoint and Trace (DWT)

The DWT contains four comparators which can be configured to generate:

- PC sampling packets at set intervals
- PC or Data watchpoint packets
- Watchpoint event to halt core

The DWT contains counters for:

- Clock cycle (CYCCNT)
- Folded instructions
- Load Store Unit (LSU) operations
- Sleep cycles
- CPI (all instruction cycles except for the first cycle)
- Interrupt overhead

15.7.7 Instrumentation Trace Macrocell (ITM)

The ITM is an application driven trace source that supports printf style debugging to trace Operating System (OS) and application events, and emits diagnostic system information. The ITM emits trace information as packets which can be generated by three different sources with several priority levels:

- **Software trace**: Software can write directly to ITM stimulus registers. This can be done using the “printf” function. For more information, refer to Section 15.7.5 “Flash Patch Breakpoint (FPB)”.
- **Hardware trace**: The ITM emits packets generated by the DWT.
- **Time stamping**: Timestamps are emitted relative to packets. The ITM contains a 21-bit counter to generate the timestamp.

15.7.7.1 How to Configure the ITM

The following example describes how to output trace data in asynchronous trace mode.

1. Configure the TPIU for asynchronous trace mode. Refer to Section 15.7.7.3 “How to Configure the TPIU”.
2. Enable the write accesses into the ITM registers by writing “0xC5ACCE55” into the Lock Access Register (Address: 0xE0000FB0)
3. Write 0x00010015 into the Trace Control register:
   - Enable ITM.
   - Enable Synchronization packets.
   - Enable SWO behavior.
   - Fix the ATB ID to 1.
4. Write 0x1 into the Trace Enable register:
   - Enable the Stimulus port 0.
5. Write 0x1 into the Trace Privilege register:

Downloaded from Arrow.com.

- Stimulus port 0 only accessed in privileged mode (Clearing a bit in this register will result in the corresponding stimulus port being accessible in user mode.)

6. Write into the Stimulus port 0 register: TPIU (Trace Port Interface Unit)

   The TPIU acts as a bridge between the on-chip trace data and the Instruction Trace Macrocell (ITM).

   The TPIU formats and transmits trace data off-chip at frequencies asynchronous to the core.

15.7.7.2 Asynchronous Mode

The TPIU is configured in asynchronous mode, trace data are output using the single TRACESWO pin. The TRACESWO signal is multiplexed with the TDO signal. As a consequence, asynchronous trace mode is only available when the Serial Wire Debug mode is selected.

Two encoding formats are available for the single pin output:

- Manchester encoded stream. This is the reset value.
- NRZ_based UART byte structure

15.7.7.3 How to Configure the TPIU

This example only concerns the asynchronous trace mode.

1. Set the TRCENA bit to 1 into the Debug Exception and Monitor Register (0xE000EDFC) to enable the use of trace and debug blocks.
2. Write 0x2 into the Selected Pin Protocol Register.
   - Select the Serial Wire output – NRZ
3. Write 0x100 into the Formatter and Flush Control Register.
4. Set the suitable clock prescaler value into the Async Clock Prescaler Register to scale the baud rate of the asynchronous output (this can be done automatically by the debugging tool).

15.7.8 IEEE1149.1 JTAG Boundary Scan

IEEE1149.1 JTAG Boundary Scan allows pin-level access independent of the device packaging technology.

IEEE1149.1 JTAG Boundary Scan is enabled when TST is tied to high, PD0 tied to low, and JTAGSEL tied to high during power-up. These pins must be maintained in their respective states for the duration of the boundary scan operation. The SAMPLE, EXTEST and BYPASS functions are implemented. In Serial Wire Debug mode, the ARM processor responds with a non-JTAG chip ID that identifies the processor. This is not IEEE1149.1 JTAG-compliant.

It is not possible to switch directly between JTAG Boundary Scan and SWJ Debug Port operations. A chip reset must be performed after JTAGSEL is changed.

A Boundary Scan Descriptor Language (BSDL) file to set up the test is provided on www.atmel.com.

15.7.8.1 JTAG Boundary Scan Register

The Boundary Scan Register (BSR) contains a number of bits which correspond to active pins and associated control signals.

Each input/output pin corresponds to a 3-bit register in the BSR. The OUTPUT bit contains data that can be forced on the pad. The INPUT bit facilitates the observability of data applied to the pad. The CONTROL bit selects the direction of the pad.

For more information, refer to BDSL files available on www.atmel.com.

Downloaded from Arrow.com.

15.7.9 ID Code Register

Access: Read-only

31	30	29	28	27	26	25	24
VERSION				PART NUMBER			

23	22	21	20	19	18	17	16
PART NUMBER							

15	14	13	12	11	10	9	8
PART NUMBER				MANUFACTURER IDENTITY			

7	6	5	4	3	2	1	0
MANUFACTURER IDENTITY							1

- **VERSION[31:28]: Product Version Number**

Set to 0x0.

- **PART NUMBER[27:12]: Product Part Number**

PART NUMBER
0x5B3D

- **MANUFACTURER IDENTITY[11:1]**

Set to 0x01F.

- **Bit[0] Required by IEEE Std. 1149.1.**

Set to 0x1.

JTAG ID Code
0x05B3_D03F

Downloaded from Arrow.com.

16. SAM-BA Boot Program

16.1 Description

The SAM-BA Boot Program integrates an array of programs permitting download and/or upload into the different memories of the product.

16.2 Embedded Characteristics

- Default Boot Program
- Interface with SAM-BA Graphic User Interface
- SAM-BA Boot
  - Supports several communication media
    - Serial Communication on UART0
    - USB device port communication up to 1Mbyte/s
  - USB Requirements
    - External crystal or external clock with frequency of 12 MHz or 16 MHz

16.3 Hardware and Software Constraints

- SAM-BA Boot uses the first 2048 bytes of the SRAM for variables and stacks. The remaining available bytes can be used for user code.
- USB Requirements
  - External crystal or external clock(1) with frequency of 12 MHz or 16 MHz
- UART0 requirements: None. If no accurate external clock source is available, the internal 12 MHz RC meets RS232 standards.

Note: 1. Must be 2500 ppm and 1.8V square wave signal.

Table 16-1. Pins Driven during Boot Program Execution

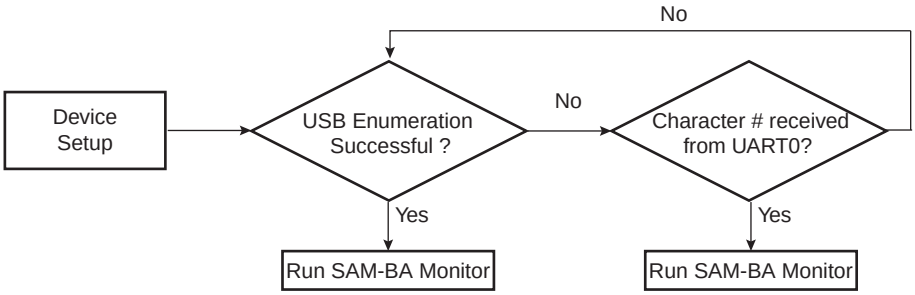
Peripheral	Pin	PIO Line
UART0	URXD0	PA9
UART0	UTXD0	PA10

Downloaded from Arrow.com.

16.4 Flow Diagram

The boot program implements the algorithm in Figure 16-1.

Figure 16-1. Boot Program Algorithm Flow Diagram



The SAM-BA boot program looks for a source clock, either from the embedded main oscillator with external crystal (main oscillator enabled) or from a supported frequency signal applied to the XIN pin (Main oscillator in bypass mode).

If a clock is supplied by one of the two sources, the boot program checks that the frequency is one of the supported external frequencies. If the frequency is supported, USB activation is allowed. If no clock is supplied, or if a clock is supplied but the frequency is not a supported external frequency, the internal 12 MHz RC oscillator is used as the main clock. In this case, the USB is not activated due to the frequency drift of the 12 MHz RC oscillator.

16.5 Device Initialization

Initialization by the boot program follows the steps described below:

1. Stack setup.
2. Embedded Flash Controller setup.
3. External clock (crystal or external clock on XIN) detection.
4. External crystal or clock with supported frequency supplied.
    1. If yes, USB activation is allowed.
    2. If no, USB activation is not allowed. The internal 12 MHz RC oscillator is used.
5. Master clock switch to main oscillator.
6. C variable initialization.
7. PLLA setup: PLLA is initialized to generate a 48 MHz clock.
8. Watchdog disable.
9. Initialization of UART0 (115200 bauds, 8, N, 1).
10. Initialization of the USB Device Port (only if USB activation is allowed—see Step 4.).
11. Wait for one of the following events:
    1. Check if USB device enumeration has occurred.
    2. Check if characters have been received in UART0.
12. Jump to SAM-BA Monitor (see Section 16.6 ”SAM-BA Monitor”)

Downloaded from Arrow.com.

16.6 SAM-BA Monitor

Once the communication interface is identified, the monitor runs in an infinite loop, waiting for different commands as shown in Table 16-2.

Table 16-2. Commands Available through the SAM-BA Boot

Command	Action	Argument(s)	Example
N	Set Normal mode	No argument	**N**#
T	Set Terminal mode	No argument	**T**#
O	Write a byte	Address, Value#	**O**200001,CA#
o	Read a byte	Address,#	**o**200001,#
H	Write a half word	Address, Value#	**H**200002,CAFE#
h	Read a half word	Address,#	**h**200002,#
W	Write a word	Address, Value#	**W**200000,CAFEDECA#
w	Read a word	Address,#	**w**200000,#
S	Send a file	Address,#	**S**200000,#
R	Receive a file	Address, NbOfBytes#	**R**200000,1234#
G	Go	Address#	**G**200200#
V	Display version	No argument	**V**#

- Mode commands:
  - Normal mode configures SAM-BA Monitor to send/receive data in binary format
  - Terminal mode configures SAM-BA Monitor to send/receive data in ASCII format
- Write commands: Write a byte (**O**), a halfword (**H**) or a word (**W**) to the target
  - *Address*: Address in hexadecimal
  - *Value*: Byte, halfword or word to write in hexadecimal
- Read commands: Read a byte (**o**), a halfword (**h**) or a word (**w**) from the target
  - *Address*: Address in hexadecimal
  - *Output*: The byte, halfword or word read in hexadecimal
- Send a file (**S**): Send a file to a specified address
  - *Address*: Address in hexadecimal

Note: There is a time-out on this command which is reached when the prompt '>' appears before the end of the command execution.

- Receive a file (**R**): Receive data into a file from a specified address
  - *Address*: Address in hexadecimal
  - *NbOfBytes*: Number of bytes in hexadecimal to receive
- Go (**G**): Jump to a specified address and execute the code
  - *Address*: Address to jump in hexadecimal
- Get Version (**V**): Return the SAM-BA boot version

Note: In Terminal mode, when the requested command is performed, SAM-BA Monitor adds the following prompt sequence to its answer: <LF>+<CR>+'>'.

Downloaded from Arrow.com.

16.6.1 UART0 Serial Port

Communication is performed through the UART0 initialized to 115200 Baud, 8, n, 1.

The Send and Receive File commands use the Xmodem protocol to communicate. Any terminal performing this protocol can be used to send the application file to the target. The size of the binary file to send depends on the SRAM size embedded in the product. In all cases, the size of the binary file must be smaller than the SRAM size because the Xmodem protocol requires some SRAM memory to work. See Section 16.3 ”Hardware and Software Constraints”

16.6.2 Xmodem Protocol

The Xmodem protocol supported is the 128-byte length block. This protocol uses a two-character CRC-16 to guarantee detection of a maximum bit error.

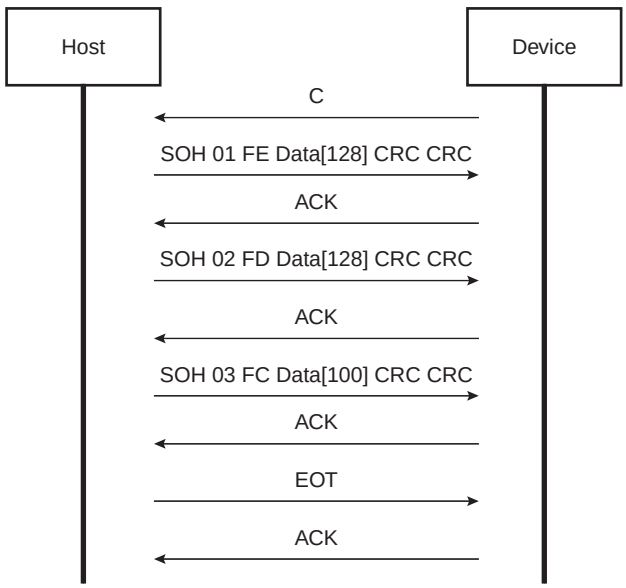
The Xmodem protocol with CRC is accurate if both sender and receiver report successful transmission. Each block of the transfer has the following format:

<SOH><blk #><255-blk #><--128 data bytes--><checksum> in which:

- <SOH> = 01 hex
- <blk #> = binary number, starts at 01, increments by 1, and wraps 0FFH to 00H (not to 01)
- <255-blk #> = 1’s complement of the blk#.
- <checksum> = 2 bytes CRC16

Figure 16-2 shows a transmission using this protocol.

Figure 16-2. Xmodem Transfer Example



Downloaded from Arrow.com.

16.6.3 USB Device Port

The device uses the USB communication device class (CDC) drivers to take advantage of the installed PC RS-232 software to talk over the USB. The CDC class is implemented in all releases of Windows®, beginning with Windows 98SE. The CDC document, available at www.usb.org, describes a way to implement devices such as ISDN modems and virtual COM ports.

The Vendor ID (VID) is the Atmel vendor ID 0x03EB. The product ID (PID) is 0x6124. These references are used by the host operating system to mount the correct driver. On Windows systems, the INF files contain the correspondence between vendor ID and product ID.

For more details on VID/PID for end product/systems, refer to the Vendor ID form available from the USB Implementers Forum found at http://www.usb.org/.

WARNING **Unauthorized use of assigned or unassigned USB Vendor ID Numbers and associated Product ID Numbers is strictly prohibited.**

16.6.3.1 Enumeration Process

The USB protocol is a master/slave protocol. This is the host that starts the enumeration sending requests to the device through the control endpoint. The device handles standard requests as defined in the USB Specification.

Table 16-3. Handled Standard Requests

Request	Definition
GET_DESCRIPTOR	Returns the current device configuration value.
SET_ADDRESS	Sets the device address for all future device access.
SET_CONFIGURATION	Sets the device configuration.
GET_CONFIGURATION	Returns the current device configuration value.
GET_STATUS	Returns status for the specified recipient.
SET_FEATURE	Set or Enable a specific feature.
CLEAR_FEATURE	Clear or Disable a specific feature.

The device also handles some class requests defined in the CDC class. .

Table 16-4. Handled Class Requests

Request	Definition
SET_LINE_CODING	Configures DTE rate, stop bits, parity and number of character bits.
GET_LINE_CODING	Requests current DTE rate, stop bits, parity and number of character bits.
SET_CONTROL_LINE_STATE	RS-232 signal used to tell the DCE device the DTE device is now present.

Unhandled requests are STALLed.

16.6.3.2 Communication Endpoints

There are two communication endpoints. Endpoint 0 is used for the enumeration process. Endpoint 1 is a 64-byte Bulk OUT endpoint. Endpoint 2 is a 64-byte Bulk IN endpoint. SAM-BA Boot commands are sent by the host through endpoint 1. If required, the message is split by the host into several data payloads by the host driver.

If the command requires a response, the host can send IN transactions to pick up the response.

Downloaded from Arrow.com.

16.6.4 In Application Programming (IAP) Feature

The IAP feature is a function located in ROM that can be called by any software application.

When called, this function sends the desired FLASH command to the EEFC and waits for the Flash to be ready (looping while the FRDY bit is not set in the MC_FSR register).

Since this function is executed from ROM, this allows Flash programming (such as sector write) to be done by code running in Flash.

The IAP function entry point is retrieved by reading the NMI vector in ROM (0x00800008).

This function takes two arguments as parameters:

- the index of the Flash bank to be programmed: 0 for EEFC0, 1 for EEFC1. For devices with only one bank, this parameter has no effect and can be either 0 or 1, only EEFC0 will be accessed.
- the command to be sent to the EEFC Command register.

This function returns the value of the EEFC_FSR register.

An example of IAP software code follows:

```
// Example: How to write data in page 200 of the flash memory using ROM IAP
function

flash_page_num = 200
flash_cmd = 0
flash_status = 0
eefc_index = 0   (0 for EEFC0, 1 for EEFC1)

// Initialize the function pointer (retrieve function address from NMI
vector)*/
iap_function_address = 0x00800008

// Fill the flash page buffer at address 200 with the data to be written
for i=0, i < page_size, i++ do
    flash_sector_200_address[i] = your_data[i]

// Prepare the command to be sent to the EEFC Command register: key, page number
and write command
flash_cmd = (0x5A << 24) | (flash_page_num << 8) | flash_write_command;

// Call the IAP function with the right parameters and retrieve the status in
flash_status after completion
flash_status = iap_function (eefc_index, flash_cmd);
```


Downloaded from Arrow.com.

17. Fast Flash Programming Interface (FFPI)

17.1 Description

The Fast Flash Programming Interface (FFPI) provides parallel high-volume programming using a standard gang programmer. The parallel interface is fully handshaked and the device is considered to be a standard EEPROM. Additionally, the parallel protocol offers an optimized access to all the embedded Flash functionalities.

Although the Fast Flash Programming mode is a dedicated mode for high volume programming, this mode is not designed for in-situ programming.

17.2 Embedded Characteristics

- Programming Mode for High-volume Flash Programming Using Gang Programmer
  - Offers Read and Write Access to the Flash Memory Plane
  - Enables Control of Lock Bits and General-purpose NVM Bits
  - Enables Security Bit Activation
  - Disabled Once Security Bit is Set
- Parallel Fast Flash Programming Interface
  - Provides an 16-bit Parallel Interface to Program the Embedded Flash
  - Full Handshake Protocol

Downloaded from Arrow.com.

17.3 Parallel Fast Flash Programming

17.3.1 Device Configuration

In Fast Flash Programming mode, the device is in a specific test mode. Only a certain set of pins is significant. The rest of the PIOs are used as inputs with a pull-up. The crystal oscillator is in bypass mode. Other pins must be left unconnected.

Figure 17-1. 16-bit Parallel Programming Interface

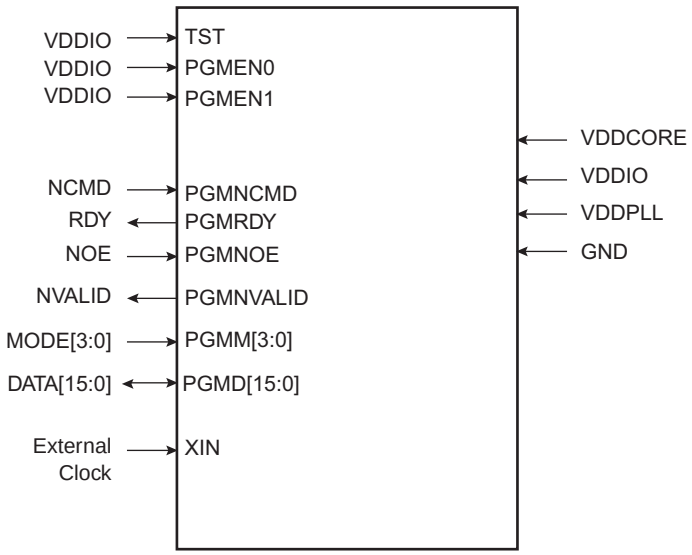


Table 17-1. Signal Description List

Signal Name	Function	Type	Active Level	Comments
Power				
VDDIO	I/O Lines Power Supply	Power	–	–
VDDCORE	Core Power Supply	Power	–	–
VDDPLL	PLL Power Supply	Power	–	–
GND	Ground	Ground	–	–
Clocks				
XIN	Main Clock Input	Input	–	–
Test				
TST	Test Mode Select	Input	High	Must be connected to VDDIO
PGMEN0	Test Mode Select	Input	High	Must be connected to VDDIO
PGMEN1	Test Mode Select	Input	High	Must be connected to VDDIO
PIO				
PGMNCMD	Valid command available	Input	Low	Pulled-up input at reset
PGMRDY	0: Device is busy 1: Device is ready for a new command	Output	High	Pulled-up input at reset
PGMNOE	Output Enable (active high)	Input	Low	Pulled-up input at reset

Downloaded from Arrow.com.

Table 17-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Comments
PGMNVALID	0: DATA[15:0] is in input mode 1: DATA[15:0] is in output mode	Output	Low	Pulled-up input at reset
PGMM[3:0]	Specifies DATA type (see Table 17-2)	Input	–	Pulled-up input at reset
PGMD[15:0]	Bi-directional data bus	Input/Output	–	Pulled-up input at reset

17.3.2 Signal Names

Depending on the MODE settings, DATA is latched in different internal registers.

Table 17-2. Mode Coding

MODE[3:0]	Symbol	Data
0000	CMDE	Command Register
0001	ADDR0	Address Register LSBs
0010	ADDR1	–
0011	ADDR2	–
0100	ADDR3	Address Register MSBs
0101	DATA	Data Register
Default	IDLE	No register

When MODE is equal to CMDE, then a new command (strobed on DATA[15:0] signals) is stored in the command register.

Table 17-3. Command Bit Coding

DATA[15:0]	Symbol	Command Executed
0x0011	READ	Read Flash
0x0012	WP	Write Page Flash
0x0022	WPL	Write Page and Lock Flash
0x0032	EWP	Erase Page and Write Page
0x0042	EWPL	Erase Page and Write Page then Lock
0x0013	EA	Erase All
0x0014	SLB	Set Lock Bit
0x0024	CLB	Clear Lock Bit
0x0015	GLB	Get Lock Bit
0x0034	SGPB	Set General Purpose NVM bit
0x0044	CGPB	Clear General Purpose NVM bit
0x0025	GGPB	Get General Purpose NVM bit
0x0054	SSE	Set Security Bit
0x0035	GSE	Get Security Bit
0x001F	WRAM	Write Memory
0x001E	GVE	Get Version

Downloaded from Arrow.com.

17.3.3 Entering Parallel Programming Mode

The following algorithm puts the device in Parallel Programming mode:

1. Apply the supplies as described in Table 17-1.
2. If an external clock is available, apply it to XIN within the VDDCORE POR reset time-out period, as defined in the section “Electrical Characteristics”.
3. Wait for the end of this reset period.
4. Start a read or write handshaking.

17.3.4 Programmer Handshaking

A handshake is defined for read and write operations. When the device is ready to start a new operation (RDY signal set), the programmer starts the handshake by clearing the NCMD signal. The handshaking is completed once the NCMD signal is high and RDY is high.

17.3.4.1 Write Handshaking

For details on the write handshaking sequence, refer to Figure 17-2 and Table 17-4.

Figure 17-2. Parallel Programming Timing, Write Sequence

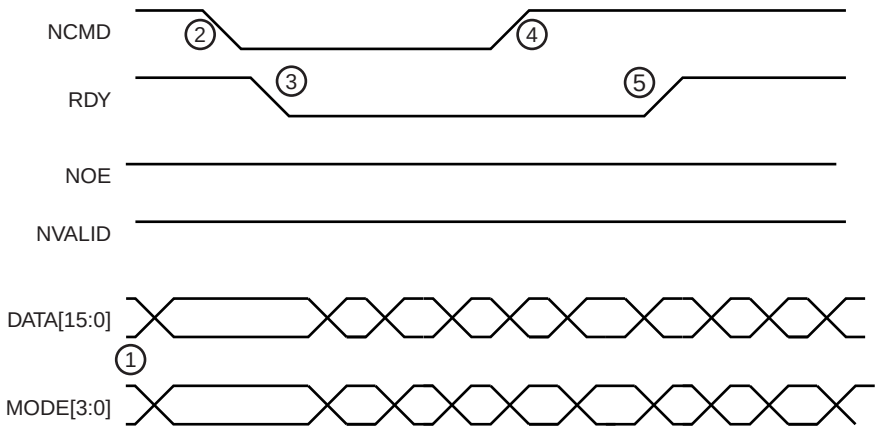


Table 17-4. Write Handshake

Step	Programmer Action	Device Action	Data I/O
1	Sets MODE and DATA signals	Waits for NCMD low	Input
2	Clears NCMD signal	Latches MODE and DATA	Input
3	Waits for RDY low	Clears RDY signal	Input
4	Releases MODE and DATA signals	Executes command and polls NCMD high	Input
5	Sets NCMD signal	Executes command and polls NCMD high	Input
6	Waits for RDY high	Sets RDY	Input

Downloaded from Arrow.com.

17.3.4.2 Read Handshaking

For details on the read handshaking sequence, refer to Figure 17-3 and Table 17-5.

Figure 17-3. Parallel Programming Timing, Read Sequence

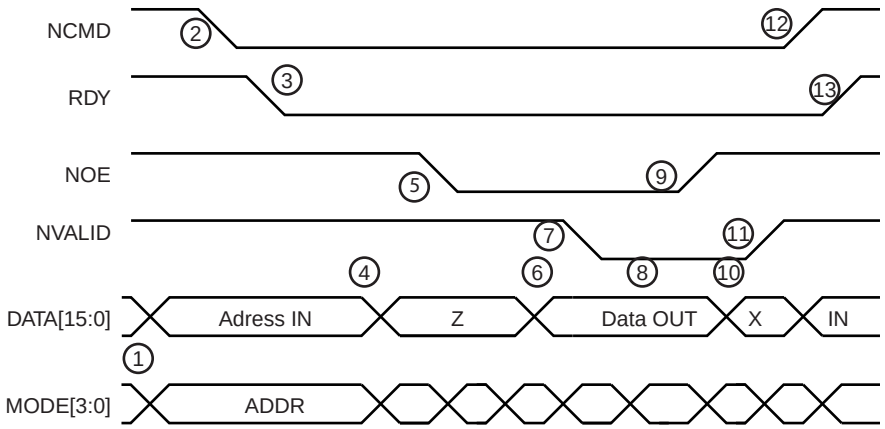


Table 17-5. Read Handshake

Step	Programmer Action	Device Action	DATA I/O
1	Sets MODE and DATA signals	Waits for NCMD low	Input
2	Clears NCMD signal	Latch MODE and DATA	Input
3	Waits for RDY low	Clears RDY signal	Input
4	Sets DATA signal in tristate	Waits for NOE Low	Input
5	Clears NOE signal	–	Tristate
6	Waits for NVALID low	Sets DATA bus in output mode and outputs the flash contents.	Output
7	–	Clears NVALID signal	Output
8	Reads value on DATA Bus	Waits for NOE high	Output
9	Sets NOE signal	–	Output
10	Waits for NVALID high	Sets DATA bus in input mode	X
11	Sets DATA in output mode	Sets NVALID signal	Input
12	Sets NCMD signal	Waits for NCMD high	Input
13	Waits for RDY high	Sets RDY signal	Input

17.3.5 Device Operations

Several commands on the Flash memory are available. These commands are summarized in Table 17-3. Each command is driven by the programmer through the parallel interface running several read/write handshaking sequences.

When a new command is executed, the previous one is automatically achieved. Thus, chaining a read command after a write automatically flushes the load buffer in the Flash.

Downloaded from Arrow.com.

17.3.5.1 Flash Read Command

This command is used to read the contents of the Flash memory. The read command can start at any valid address in the memory plane and is optimized for consecutive reads. Read handshaking can be chained; an internal address buffer is automatically increased.

Table 17-6. Read Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	READ
2	Write handshaking	ADDR0	Memory Address LSB
3	Write handshaking	ADDR1	Memory Address
4	Read handshaking	DATA	*Memory Address++
5	Read handshaking	DATA	*Memory Address++
...	...	...	...
n	Write handshaking	ADDR0	Memory Address LSB
n+1	Write handshaking	ADDR1	Memory Address
n+2	Read handshaking	DATA	*Memory Address++
n+3	Read handshaking	DATA	*Memory Address++
...	...	...	...

17.3.5.2 Flash Write Command

This command is used to write the Flash contents.

The Flash memory plane is organized into several pages. Data to be written are stored in a load buffer that corresponds to a Flash memory page. The load buffer is automatically flushed to the Flash:

- before access to any page other than the current one
- when a new command is validated (MODE = CMDE)

The **Write Page** command **(WP)** is optimized for consecutive writes. Write handshaking can be chained; an internal address buffer is automatically increased.

Table 17-7. Write Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	WP or WPL or EWP or EWPL
2	Write handshaking	ADDR0	Memory Address LSB
3	Write handshaking	ADDR1	Memory Address
4	Write handshaking	DATA	*Memory Address++
5	Write handshaking	DATA	*Memory Address++
...	...	...	...
n	Write handshaking	ADDR0	Memory Address LSB
n+1	Write handshaking	ADDR1	Memory Address
n+2	Write handshaking	DATA	*Memory Address++
n+3	Write handshaking	DATA	*Memory Address++
...	...	...	...

Downloaded from Arrow.com.

The Flash command **Write Page and Lock (WPL)** is equivalent to the Flash Write Command. However, the lock bit is automatically set at the end of the Flash write operation. As a lock region is composed of several pages, the programmer writes to the first pages of the lock region using Flash write commands and writes to the last page of the lock region using a Flash write and lock command.

The Flash command **Erase Page and Write (EWP)** is equivalent to the Flash Write Command. However, before programming the load buffer, the page is erased.

The Flash command **Erase Page and Write the Lock (EWPL)** combines EWP and WPL commands.

17.3.5.3 Flash Full Erase Command

This command is used to erase the Flash memory planes.

All lock regions must be unlocked before the Full Erase command by using the CLB command. Otherwise, the erase command is aborted and no page is erased.

Table 17-8. Full Erase Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	EA
2	Write handshaking	DATA	0

17.3.5.4 Flash Lock Commands

Lock bits can be set using WPL or EWPL commands. They can also be set by using the **Set Lock** command **(SLB)**. With this command, several lock bits can be activated. A Bit Mask is provided as argument to the command. When bit 0 of the bit mask is set, then the first lock bit is activated.

In the same way, the **Clear Lock** command **(CLB)** is used to clear lock bits.

Table 17-9. Set and Clear Lock Bit Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	SLB or CLB
2	Write handshaking	DATA	Bit Mask

Lock bits can be read using **Get Lock Bit** command **(GLB)**. The nth lock bit is active when the bit n of the bit mask is set.

Table 17-10. Get Lock Bit Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	GLB
2	Read handshaking	DATA	Lock Bit Mask Status 0 = Lock bit is cleared 1 = Lock bit is set

Downloaded from Arrow.com.

17.3.5.5 Flash General-purpose NVM Commands

General-purpose NVM bits (GP NVM bits) can be set using the **Set GPNVM** command **(SGPB)**. This command also activates GP NVM bits. A bit mask is provided as argument to the command. When bit 0 of the bit mask is set, then the first GP NVM bit is activated.

In the same way, the **Clear GPNVM** command **(CGPB)** is used to clear general-purpose NVM bits. The general-purpose NVM bit is deactivated when the corresponding bit in the pattern value is set to 1.

Table 17-11. Set/Clear GP NVM Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	SGPB or CGPB
2	Write handshaking	DATA	GP NVM bit pattern value

General-purpose NVM bits can be read using the **Get GPNVM Bit** command **(GGPB)**. The nth GP NVM bit is active when bit n of the bit mask is set.

Table 17-12. Get GP NVM Bit Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	GGPB
2	Read handshaking	DATA	GP NVM Bit Mask Status 0 = GP NVM bit is cleared 1 = GP NVM bit is set

17.3.5.6 Flash Security Bit Command

A security bit can be set using the **Set Security Bit** command (SSE). Once the security bit is active, the Fast Flash programming is disabled. No other command can be run. An event on the Erase pin can erase the security bit once the contents of the Flash have been erased.

Table 17-13. Set Security Bit Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	SSE
2	Write handshaking	DATA	0

Once the security bit is set, it is not possible to access FFPI. The only way to erase the security bit is to erase the Flash.

To erase the Flash, perform the following steps:

1. Power-off the chip.
2. Power-on the chip with TST = 0.
3. Assert the ERASE pin for at least the ERASE pin assertion time as defined in the section “Electrical Characteristics”.
4. Power-off the chip.

Return to FFPI mode to check that the Flash is erased.

Downloaded from Arrow.com.

17.3.5.7 Memory Write Command

This command is used to perform a write access to any memory location.

The **Memory Write** command **(WRAM)** is optimized for consecutive writes. Write handshaking can be chained; an internal address buffer is automatically increased.

Table 17-14. Write Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	WRAM
2	Write handshaking	ADDR0	Memory Address LSB
3	Write handshaking	ADDR1	Memory Address
4	Write handshaking	DATA	*Memory Address++
5	Write handshaking	DATA	*Memory Address++
...	...	...	...
n	Write handshaking	ADDR0	Memory Address LSB
n+1	Write handshaking	ADDR1	Memory Address
n+2	Write handshaking	DATA	*Memory Address++
n+3	Write handshaking	DATA	*Memory Address++
...	...	...	...

17.3.5.8 Get Version Command

The **Get Version** (GVE) command retrieves the version of the FFPI interface.

Table 17-15. Get Version Command

Step	Handshake Sequence	MODE[3:0]	DATA[15:0]
1	Write handshaking	CMDE	GVE
2	Read handshaking	DATA	Version

Downloaded from Arrow.com.

18. Bus Matrix (MATRIX)

18.1 Description

The Bus Matrix (MATRIX) implements a multi-layer AHB, based on the AHB-Lite protocol, that enables parallel access paths between multiple AHB masters and slaves in a system, thus increasing the overall bandwidth. The Bus Matrix interconnects 12 AHB masters to 9 AHB slaves. The normal latency to connect a master to a slave is one cycle. The exception is the default master of the accessed slave which is connected directly (zero cycle latency).

The Bus Matrix user interface is compliant with ARM Advanced Peripheral Bus.

18.2 Embedded Characteristics

- 12 Masters
- 9 Slaves
- One Decoder for Each Master
- Several Possible Boot Memories for Each Master before Remap
- One Remap Function for Each Master
- Support for Long Bursts of 32, 64, 128 and up to the 256-beat Word Burst AHB Limit
- Enhanced Programmable Mixed Arbitration for Each Slave
  - Round-Robin
  - Fixed Priority
- Programmable Default Master for Each Slave
  - No Default Master
  - Last Accessed Default Master
  - Fixed Default Master
- Deterministic Maximum Access Latency for Masters
- Zero or One Cycle Arbitration Latency for the First Access of a Burst
- Bus Lock Forwarding to Slaves
- Master Number Forwarding to Slaves
- One Special Function Register for Each Slave (not dedicated)
- Register Write Protection

Downloaded from Arrow.com.

18.2.1 Matrix Masters

The Bus Matrix manages the masters listed in Table 18-1. Each master can perform an access to an available slave concurrently with other masters. lists the available masters.

Each master has its own specifically-defined decoder. To simplify addressing, all the masters have the same decodings.

Table 18-1. Bus Matrix Masters

Master Index	Name
0	Cortex-M7
1	Cortex-M7
2	Cortex-M7 Peripheral Port
3	Integrated Check Monitor
4, 5	XDMAC
6	ISI DMA
7	Reserved
8	USB DMA
9	Ethernet MAC DMA
10	CAN0 DMA
11	CAN1 DMA

18.2.2 Matrix Slaves

The Bus Matrix manages the slaves listed in Table 18-2. Each slave has its own arbiter, providing a different arbitration per slave.

Table 18-2. Bus Matrix Slaves

Slave Index	Name
0	Internal SRAM
1	Internal SRAM
2	Internal ROM
3	Internal Flash
4	USB High Speed Dual Port RAM (DPR)
5	External Bus Interface
6	QSPI
7	Peripheral Bridge
8	AHB Slave

Downloaded from Arrow.com.

18.2.3 Master to Slave Access

Table 18-3 provides valid paths for master to slave accesses. The paths shown as "-" are forbidden or not wired.

Table 18-3. Master to Slave Access

Masters		0	1	2	3	4	5	6	7	8	9	10	11
Slaves		Cortex-M7	Cortex-M7	Cortex-M7 Periph. Port	ICM	Central DMA IF0	Central DMA IF1	ISI DMA	Reserved	USB DMA	GMAC DMA	CAN0 DMA	CAN1 DMA
0	Internal SRAM	–	–	–	X	X	–	–	–	–	–	–	–
1	Internal SRAM	–	–	–	–	–	X	X	–	X	X	X	X
2	Internal ROM	X	–	–	–	–	–	–	–	–	–	–	–
3	Internal Flash	X	–	–	X	–	X	–	–	X	X	–	–
4	USB High-speed Dual Port RAM	–	X	–	–	–	–	–	–	–	–	–	–
5	External Bus Interface	–	X	–	X	X	X	X	–	X	X	X	X
6	QSPI	X	–	–	X	–	X	–	–	X	X	–	–
7	Peripheral Bridge	–	X	X	–	–	X	–	–	–	–	–	–
8	Cortex-M7 AHB Slave (AHBS)[1]	–	–	–	X	X	–	X	–	X	X	X	X

Note: 1. The connection of the Cortex-M7 processor to the SRAM is defined in the sections "Interconnect" and "Memories", sub-section "Embedded Memories", of this datasheet.

Downloaded from Arrow.com.

18.3 Functional Description

18.3.1 Memory Mapping

The Bus Matrix provides one decoder for every AHB master interface. The decoder offers each AHB master several memory mappings. Each memory area may be assigned to several slaves. Thus booting at the same address while using different AHB slaves (i.e., external RAM, internal ROM or internal Flash, etc.) is possible.

The Bus Matrix user interface provides the Master Remap Control Register (MATRIX_MRCR) that performs remap action for every master independently.

18.3.2 Special Bus Granting Mechanism

The Bus Matrix provides some speculative bus granting techniques in order to anticipate access requests from masters. This technique reduces latency at the first access of a burst, or for a single transfer, as long as the slave is free from any other master access. Bus granting sets a different default master for every slave.

At the end of the current access, if no other request is pending, the slave remains connected to its associated default master. A slave can be associated with three kinds of default masters:

- No default master
- Last access master
- Fixed default master

To change from one type of default master to another, the Bus Matrix user interface provides the Slave Configuration registers, one for every slave, that set a default master for each slave. The Slave Configuration register contains the fields DEFMSTR_TYPE and FIXED_DEFMSTR. The 2-bit DEFMSTR_TYPE field selects the default master type (no default, last access master, fixed default master), whereas the 4-bit FIXED_DEFMSTR field selects a fixed default master provided that DEFMSTR_TYPE is set to fixed default master. Refer to Section 18.4.2 ”Bus Matrix Slave Configuration Registers”.

18.3.2.1 No Default Master

After the end of the current access, if no other request is pending, the slave is disconnected from all masters.

This configuration incurs one latency clock cycle for the first access of a burst after bus Idle. Arbitration without default master may be used for masters that perform significant bursts or several transfers with no Idle in between, or if the slave bus bandwidth is widely used by one or more masters.

This configuration provides no benefit on access latency or bandwidth when reaching maximum slave bus throughput whatever the number of requesting masters.

18.3.2.2 Last Access Master

After the end of the current access, if no other request is pending, the slave remains connected to the last master that performed an access request.

This allows the Bus Matrix to remove the one latency cycle for the last master that accessed the slave. Other non privileged masters still get one latency clock cycle if they want to access the same slave. This technique is useful for masters that mainly perform single accesses or short bursts with some Idle cycles in between.

This configuration provides no benefit on access latency or bandwidth when reaching maximum slave bus throughput whatever is the number of requesting masters.

18.3.2.3 Fixed Default Master

At the end of the current access, if no other request is pending, the slave connects to its fixed default master. Unlike the last access master, the fixed default master does not change unless the user modifies it by software (FIXED_DEFMSTR field of the related MATRIX_SCFG).

This allows the Bus Matrix arbiters to remove the one latency clock cycle for the fixed default master of the slave. All requests attempted by the fixed default master do not cause any arbitration latency, whereas other non-

Downloaded from Arrow.com.

privileged masters will get one latency cycle. This technique is useful for a master that mainly performs single accesses or short bursts with Idle cycles in between.

This configuration provides no benefit on access latency or bandwidth when reaching maximum slave bus throughput, regardless of the number of requesting masters.

18.3.3 Arbitration

The Bus Matrix provides an arbitration technique that reduces latency when conflicting cases occur; for example. when two or more masters try to access the same slave at the same time. One arbiter per AHB slave is provided, so that each slave is arbitrated differently.

The Bus Matrix provides the user with two arbitration types for each slave:

1. Round-robin Arbitration (default)
2. Fixed Priority Arbitration

Each algorithm may be complemented by selecting a default master configuration for each slave.

When re-arbitration is required, specific conditions apply. See Section 18.3.3.1 ”Arbitration Rules”.

18.3.3.1 Arbitration Rules

Each arbiter has the ability to arbitrate between requests from two or more masters. To avoid burst breaking and to provide maximum throughput for slave interfaces, arbitration should take place during the following cycles:

1. Idle cycles: When a slave is not connected to any master or is connected to a master which is not currently accessing it.
2. Single cycles: When a slave is performing a single access.
3. End of Burst cycles: When the current cycle is the last cycle of a burst transfer. For a defined length burst, predicted end of burst matches the size of the transfer but is managed differently for undefined length burst. See Section ”Undefined Length Burst Arbitration”
4. Slot cycle limit: When the slot cycle counter has reached the limit value indicating that the current master access is too long and must be broken. See Section ”Slot Cycle Limit Arbitration”

Undefined Length Burst Arbitration

In order to prevent slave handling during undefined length bursts, the user can trigger the re-arbitration before the end of the incremental bursts. The re-arbitration period can be selected from the following Undefined Length Burst Type (ULBT) possibilities:

1. Unlimited: no predetermined end of burst is generated. This value enables 1-Kbyte burst lengths.
2. 1-beat bursts: predetermined end of burst is generated at each single transfer during the INCR transfer.
3. 4-beat bursts: predetermined end of burst is generated at the end of each 4-beat boundary during INCR transfer.
4. 8-beat bursts: predetermined end of burst is generated at the end of each 8-beat boundary during INCR transfer.
5. 16-beat bursts: predetermined end of burst is generated at the end of each 16-beat boundary during INCR transfer.
6. 32-beat bursts: predetermined end of burst is generated at the end of each 32-beat boundary during INCR transfer.
7. 64-beat bursts: predetermined end of burst is generated at the end of each 64-beat boundary during INCR transfer.
8. 128-beat bursts: predetermined end of burst is generated at the end of each 128-beat boundary during INCR transfer.

The use of undefined length16-beat bursts, or less, is discouraged since this decreases the overall bus bandwidth due to arbitration and slave latencies at each first access of a burst.

Downloaded from Arrow.com.

If the master does not permanently and continuously request the same slave or has an intrinsically limited average throughput, the ULBT should be left at its default unlimited value, knowing that the AHB specification natively limits all word bursts to 256 beats and double-word bursts to 128 beats because of its 1-Kbyte address boundaries.

Unless duly needed, the ULBT should be left at its default value of 0 for power saving.

This selection is made through the ULBT field of the Master Configuration Registers (MATRIX_MCFG).

Slot Cycle Limit Arbitration

The Bus Matrix contains specific logic to break long accesses, such as very long bursts on a very slow slave (e.g., an external low speed memory). At each arbitration time, a counter is loaded with the value previously written in the SLOT_CYCLE field of the related Slave Configuration Register (MATRIX_SCFG) and decreased at each clock cycle. When the counter elapses, the arbiter has the ability to re-arbitrate at the end of the current AHB bus access cycle.

Unless a master has a very tight access latency constraint, which could lead to data overflow or underflow due to a badly undersized internal FIFO with respect to its throughput, the Slot Cycle Limit should be disabled (SLOT_CYCLE = 0) or set to its default maximum value in order not to inefficiently break long bursts performed by some Atmel masters.

In most cases, this feature is not needed and should be disabled for power saving.

WARNING: This feature does not prevent a slave from locking its access indefinitely.

18.3.3.2 Arbitration Priority Scheme

The Bus Matrix arbitration scheme is organized in priority pools.

Round-robin priority is used in the highest and lowest priority pools, whereas fixed level priority is used between priority pools and in the intermediate priority pools.

For each slave, each master is assigned to one of the slave priority pools through the priority registers for slaves (MxPR fields of MATRIX_PRAS and MATRIX_PRBS). When evaluating master requests, this programmed priority level always takes precedence.

After reset, all the masters except those of the Cortex-M7 belong to the lowest priority pool (MxPR = 0) and are therefore granted bus access in a true round-robin order.

The highest priority pool must be specifically reserved for masters requiring very low access latency. If more than one master belongs to this pool, they will be granted bus access in a biased round-robin manner which allows tight and deterministic maximum access latency from AHB bus requests. In the worst case, any currently occurring high-priority master request will be granted after the current bus master access has ended and other high priority pool master requests, if any, have been granted once each.

The lowest priority pool shares the remaining bus bandwidth between AHB Masters.

Intermediate priority pools allow fine priority tuning. Typically, a moderately latency-critical master or a bandwidth-only critical master will use such a priority level. The higher the priority level (MxPR value), the higher the master priority.

All combinations of MxPR values are allowed for all masters and slaves. For example, some masters might be assigned the highest priority pool (round-robin), and remaining masters the lowest priority pool (round-robin), with no master for intermediate fix priority levels.

If more than one master requests the slave bus, regardless of the respective masters priorities, no master will be granted the slave bus for two consecutive runs. A master can only get back-to-back grants so long as it is the only requesting master.

Fixed Priority Arbitration

The fixed priority arbitration algorithm is the first and only arbitration algorithm applied between masters from distinct priority pools. It is also used in priority pools other than the highest and lowest priority pools (intermediate priority pools).

Downloaded from Arrow.com.

Fixed priority arbitration is used by the Bus Matrix arbiters to dispatch the requests from different masters to the same slave by using the fixed priority defined by the user. If requests from two or more masters are active at the same time, the master with the highest priority number is serviced first. If requests from two or more masters with the same priority are active at the same time, the master with the highest number is serviced first.

For each slave, the priority of each master is defined in the MxPR field in the Priority Registers, MATRIX_PRAS and MATRIX_PRBS.

Round-Robin Arbitration

Round-robin arbitration is only used in the highest and lowest priority pools. It allows the Bus Matrix arbiters to properly dispatch requests from different masters to the same slave. If two or more master requests are active at the same time in the priority pool, they are serviced in a round-robin increasing master number order.

18.3.4 System I/O Configuration

The System I/O Configuration register (CCFG_SYSIO) configures I/O lines in System I/O mode (such as JTAG, ERASE, USB, etc.) or as general purpose I/O lines. Enabling or disabling the corresponding I/O lines in peripheral mode or in PIO mode (PIO_PER or PIO_PDR registers) in the PIO controller as no effect. However, the direction (input or output), pull-up, pull-down and other mode control is still managed by the PIO controller.

18.3.5 SMC NAND Flash Chip Select Configuration

The SMC Nand Flash Chip Select Configuration Register (CCFG_SMCNFCS) manages the chip select signal (NCSx) and its assignment to NAND Flash.

Each NCSx may or may not be individually assigned to NAND Flash. When the NCSx is assigned to NAND Flash, the signals NANDOE and NANDWE are used for the NCSx signals selected.

Downloaded from Arrow.com.

18.3.6 Register Write Protection

To prevent any single software error from corrupting MATRIX behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the Write Protection Mode Register (MATRIX_WPMR).

If a write access to a write-protected register is detected, the WPVS flag in the Write Protection Status Register (MATRIX_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS flag is reset by writing the Bus Matrix Write Protect Mode Register (MATRIX_WPMR) with the appropriate access key WPKEY.

The following registers can be write-protected:

- Bus Matrix Master Configuration Registers
- Bus Matrix Slave Configuration Registers
- Bus Matrix Priority Registers A For Slaves
- Bus Matrix Priority Registers B For Slaves
- Bus Matrix Master Remap Control Register

Downloaded from Arrow.com.

18.4 Bus Matrix (MATRIX) User Interface

Table 18-4. Register Mapping

Offset	Register	Name	Access	Reset
0x0000	Master Configuration Register 0	MATRIX_MCFG0	Read/Write	0x00000001
0x0004	Master Configuration Register 1	MATRIX_MCFG1	Read/Write	0x00000000
0x0008	Master Configuration Register 2	MATRIX_MCFG2	Read/Write	0x00000000
0x000C	Master Configuration Register 3	MATRIX_MCFG3	Read/Write	0x00000000
0x0010	Master Configuration Register 4	MATRIX_MCFG4	Read/Write	0x00000000
0x0014	Master Configuration Register 5	MATRIX_MCFG5	Read/Write	0x00000000
0x0018	Master Configuration Register 6	MATRIX_MCFG6	Read/Write	0x00000000
0x001C	Reserved	–	–	–
0x0020	Master Configuration Register 8	MATRIX_MCFG8	Read/Write	0x00000000
0x0024	Master Configuration Register 9	MATRIX_MCFG9	Read/Write	0x00000000
0x0028	Master Configuration Register 10	MATRIX_MCFG10	Read/Write	0x00000000
0x002C	Master Configuration Register 11	MATRIX_MCFG11	Read/Write	0x00000000
0x0030–0x003C	Reserved	–	–	–
0x0040	Slave Configuration Register 0	MATRIX_SCFG0	Read/Write	0x000001FF
0x0044	Slave Configuration Register 1	MATRIX_SCFG1	Read/Write	0x000001FF
0x0048	Slave Configuration Register 2	MATRIX_SCFG2	Read/Write	0x000001FF
0x004C	Slave Configuration Register 3	MATRIX_SCFG3	Read/Write	0x000001FF
0x0050	Slave Configuration Register 4	MATRIX_SCFG4	Read/Write	0x000001FF
0x0054	Slave Configuration Register 5	MATRIX_SCFG5	Read/Write	0x000001FF
0x0058	Slave Configuration Register 6	MATRIX_SCFG6	Read/Write	0x000001FF
0x005C	Slave Configuration Register 7	MATRIX_SCFG7	Read/Write	0x000001FF
0x0060	Slave Configuration Register 8	MATRIX_SCFG8	Read/Write	0x000001FF
0x0064–0x007C	Reserved	–	–	–
0x0080	Priority Register A for Slave 0	MATRIX_PRAS0	Read/Write	0x00000222(1)
0x0084	Priority Register B for Slave 0	MATRIX_PRBS0	Read/Write	0x00000000(1)
0x0088	Priority Register A for Slave 1	MATRIX_PRAS1	Read/Write	0x00000222(1)
0x008C	Priority Register B for Slave 1	MATRIX_PRBS1	Read/Write	0x00000000(1)
0x0090	Priority Register A for Slave 2	MATRIX_PRAS2	Read/Write	0x00000222(1)
0x0094	Priority Register B for Slave 2	MATRIX_PRBS2	Read/Write	0x00000000(1)
0x0098	Priority Register A for Slave 3	MATRIX_PRAS3	Read/Write	0x00000222(1)
0x009C	Priority Register B for Slave 3	MATRIX_PRBS3	Read/Write	0x00000000(1)
0x00A0	Priority Register A for Slave 4	MATRIX_PRAS4	Read/Write	0x00000222(1)
0x00A4	Priority Register B for Slave 4	MATRIX_PRBS4	Read/Write	0x00000000(1)
0x00A8	Priority Register A for Slave 5	MATRIX_PRAS5	Read/Write	0x00000222(1)
0x00AC	Priority Register B for Slave 5	MATRIX_PRBS5	Read/Write	0x00000000(1)

Downloaded from Arrow.com.

Table 18-4. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x00B0	Priority Register A for Slave 6	MATRIX_PRAS6	Read/Write	0x00000222(1)
0x00B4	Priority Register B for Slave 6	MATRIX_PRBS6	Read/Write	0x00000000(1)
0x00B8	Priority Register A for Slave 7	MATRIX_PRAS7	Read/Write	0x00000222(1)
0x00BC	Priority Register B for Slave 7	MATRIX_PRBS7	Read/Write	0x00000000(1)
0x00C0	Priority Register A for Slave 8	MATRIX_PRAS8	Read/Write	0x00000222(1)
0x00C4	Priority Register B for Slave 8	MATRIX_PRBS8	Read/Write	0x00000000(1)
0x00C8–0x00FC	Reserved	–	–	–
0x0100	Master Remap Control Register	MATRIX_MRCR	Read/Write	0x00000000
0x0104–0x010C	Reserved	–	–	–
0x0110	CAN0 Configuration Register	CCFG_CAN0	Read/Write	0x2040019D
0x0114	System I/O and CAN1 Configuration Register	CCFG_SYSIO	Read/Write	0x20400000
0x0118–0x0120	Reserved	–	–	–
0x0124	SMC NAND Flash Chip Select Configuration Register	CCFG_SMCNFCS	Read/Write	0x00000000
0x0128–0x01E0	Reserved	–	–	–
0x01E4	Write Protection Mode Register	MATRIX_WPMR	Read/Write	0x00000000
0x01E8	Write Protection Status Register	MATRIX_WPSR	Read-only	0x00000000
0x01EC–0x01FC	Reserved	–	–	–

Notes: 1. Values in the Bus Matrix Priority Registers are product-dependent.

Downloaded from Arrow.com.

18.4.1 Bus Matrix Master Configuration Registers

Name: MATRIX_MCFG0..MATRIX_MCFG11

Address: 0x40088000 [0], 0x40088004 [1], 0x40088008 [2], 0x4008800C [3], 0x40088010 [4], 0x40088014 [5], 0x40088018 [6], 0x40088020 [8], 0x40088024 [9], 0x40088028 [10], 0x4008802C [11]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	ULBT		

This register can only be written if the WPEN bit is cleared in the Write Protection Mode Register.

- **ULBT: Undefined Length Burst Type**

Value	Name	Description
0	UNLTD_LENGTH	Unlimited Length Burst—No predicted end of burst is generated, therefore INCR bursts coming from this master can only be broken if the Slave Slot Cycle Limit is reached. If the Slot Cycle Limit is not reached, the burst is normally completed by the master, at the latest, on the next AHB 1-Kbyte address boundary, allowing up to 256-beat word bursts or 128-beat double-word bursts. This value should not be used in the very particular case of a master capable of performing back-to-back undefined length bursts on a single slave, since this could indefinitely freeze the slave arbitration and thus prevent another master from accessing this slave.
1	SINGLE_ACCESS	Single Access—The undefined length burst is treated as a succession of single accesses, allowing re-arbitration at each beat of the INCR burst or bursts sequence.
2	4BEAT_BURST	4-beat Burst—The undefined length burst or bursts sequence is split into 4-beat bursts or less, allowing re-arbitration every 4 beats.
3	8BEAT_BURST	8-beat Burst—The undefined length burst or bursts sequence is split into 8-beat bursts or less, allowing re-arbitration every 8 beats.
4	16BEAT_BURST	16-beat Burst—The undefined length burst or bursts sequence is split into 16-beat bursts or less, allowing re-arbitration every 16 beats.
5	32BEAT_BURST	32-beat Burst —The undefined length burst or bursts sequence is split into 32-beat bursts or less, allowing re-arbitration every 32 beats.
6	64BEAT_BURST	64-beat Burst—The undefined length burst or bursts sequence is split into 64-beat bursts or less, allowing re-arbitration every 64 beats.
7	128BEAT_BURST	128-beat Burst—The undefined length burst or bursts sequence is split into 128-beat bursts or less, allowing re-arbitration every 128 beats.

Note: Unless duly needed, the ULBT should be left at its default ‘0’ value for power saving.

Downloaded from Arrow.com.

18.4.2 Bus Matrix Slave Configuration Registers

Name: MATRIX_SCFG0..MATRIX_SCFG8

Address: 0x40088040[0], 0x40088044 [1], 0x40088048 [2], 0x4008804C [3], 0x40088050 [4], 0x40088054 [5], 0x40088058 [6], 0x4008805C [7], 0x40088060 [8]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	FIXED_DEFMSTR				DEFMSTR_TYPE	
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	SLOT_CYCLE
7	6	5	4	3	2	1	0
SLOT_CYCLE							

This register can only be written if the WPEN bit is cleared in the Write Protection Mode Register.

- **SLOT_CYCLE: Maximum Bus Grant Duration for Masters**

When SLOT_CYCLE AHB clock cycles have elapsed since the last arbitration, a new arbitration takes place to let another master access this slave. If another master is requesting the slave bus, then the current master burst is broken.

If SLOT_CYCLE = 0, the slot cycle limit feature is disabled and bursts always complete unless broken according to the ULBT.

This limit has been placed in order to enforce arbitration so as to meet potential latency constraints of masters waiting for slave access.

This limit must not be too small. Unreasonably small values break every burst and the Bus Matrix arbitrates without performing any data transfer. The default maximum value is usually an optimal conservative choice.

In most cases, this feature is not needed and should be disabled for power saving.

See “Slot Cycle Limit Arbitration” for details.

- **DEFMSTR_TYPE: Default Master Type**

Value	Name	Description
0	NONE	No Default Master—At the end of the current slave access, if no other master request is pending, the slave is disconnected from all masters. This results in a one clock cycle latency for the first access of a burst transfer or for a single access.
1	LAST	Last Default Master—At the end of the current slave access, if no other master request is pending, the slave stays connected to the last master having accessed it. This results in not having one clock cycle latency when the last master tries to access the slave again.
2	FIXED	Fixed Default Master—At the end of the current slave access, if no other master request is pending, the slave connects to the fixed master the number that has been written in the FIXED_DEFMSTR field. This results in not having one clock cycle latency when the fixed master tries to access the slave again.

- **FIXED_DEFMSTR: Fixed Default Master**

Number of the Default Master for this slave. Only used if DEFMSTR_TYPE is 2. Specifying the number of a master which is not connected to the selected slave is equivalent to setting DEFMSTR_TYPE to 0.

Downloaded from Arrow.com.

18.4.3 Bus Matrix Priority Registers A For Slaves

Name: MATRIX_PRAS0..MATRIX_PRAS8

Address: 0x40088080 [0], 0x40088088 [1], 0x40088090 [2], 0x40088098 [3], 0x400880A0 [4], 0x400880A8 [5], 0x400880B0 [6], 0x400880B8 [7], 0x400880C0 [8]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	M6PR	

23	22	21	20	19	18	17	16
–	–	M5PR		–	–	M4PR	

15	14	13	12	11	10	9	8
–	–	M3PR		–	–	M2PR	

7	6	5	4	3	2	1	0
–	–	M1PR		–	–	M0PR	

This register can only be written if the WPE bit is cleared in the Write Protection Mode Register.

- **MxPR: Master x Priority**

Fixed priority of Master x for accessing the selected slave. The higher the number, the higher the priority.

All the masters programmed with the same MxPR value for the slave make up a priority pool.

Round-robin arbitration is used in the lowest (MxPR = 0) and highest (MxPR = 3) priority pools.

Fixed priority is used in intermediate priority pools (MxPR = 1) and (MxPR = 2).

See “Arbitration Priority Scheme” for details.

Downloaded from Arrow.com.

18.4.4 Bus Matrix Priority Registers B For Slaves

Name: MATRIX_PRBS0..MATRIX_PRBS8

Address: 0x40088084 [0], 0x4008808C [1], 0x40088094 [2], 0x4008809C [3], 0x400880A4 [4], 0x400880AC [5], 0x400880B4 [6], 0x400880BC [7], 0x400880C4 [8]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	M11PR		–	–	M10PR	
7	6	5	4	3	2	1	0
–	–	M9PR		–	–	M8PR	

This register can only be written if the WPE bit is cleared in the Write Protection Mode Register.

- **MxPR: Master x Priority**

Fixed priority of Master x for accessing the selected slave. The higher the number, the higher the priority.

All the masters programmed with the same MxPR value for the slave make up a priority pool.

Round-robin arbitration is used in the lowest (MxPR = 0) and highest (MxPR = 3) priority pools.

Fixed priority is used in intermediate priority pools (MxPR = 1) and (MxPR = 2).

See “Arbitration Priority Scheme” for details.

Downloaded from Arrow.com.

18.4.5 Bus Matrix Master Remap Control Register

Name: MATRIX_MRCR

Address: 0x40088100

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RCB11	RCB10	RCB9	RCB8
7	6	5	4	3	2	1	0
–	RCB6	RCB5	RCB4	RCB3	RCB2	RCB1	RCB0

This register can only be written if the WPEN bit is cleared in the Write Protection Mode Register.

- **RCBx: Remap Command Bit for Master x**

0: Disables remapped address decoding for the selected Master.

1: Enables remapped address decoding for the selected Master.

Downloaded from Arrow.com.

18.4.6 CAN0 Configuration Register

Name: CCFG_CAN0

Address: 0x40088110

Access: Read/Write

31	30	29	28	27	26	25	24
CAN0DMABA							
23	22	21	20	19	18	17	16
CAN0DMABA							
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	Reserved
7	6	5	4	3	2	1	0
Reserved							

- **Reserved: Do not change the reset value**

- **CAN0DMABA: CAN0 DMA Base Address**

Gives the 16-bit MSB of the CAN0 DMA base address. The 16-bit LSB must be programmed into CAN0 user interface.

Default address is 0x20400000.

Downloaded from Arrow.com.

18.4.7 System I/O and CAN1 Configuration Register

Name: CCFG_SYSIO

Address: 0x40088114

Access: Read/Write

31	30	29	28	27	26	25	24
CAN1DMABA							
23	22	21	20	19	18	17	16
CAN1DMABA							
15	14	13	12	11	10	9	8
–	–	–	SYSIO12	–	–	–	–
7	6	5	4	3	2	1	0
SYSIO7	SYSIO6	SYSIO5	SYSIO4	–	–	–	–

- **SYSIO4: PB4 or TDI Assignment**

0: TDI function selected.

1: PB4 function selected.

- **SYSIO5: PB5 or TDO/TRACESWO Assignment**

0: TDO/TRACESWO function selected.

1: PB5 function selected.

- **SYSIO6: PB6 or TMS/SWDIO Assignment**

0: TMS/SWDIO function selected.

1: PB6 function selected.

- **SYSIO7: PB7 or TCK/SWCLK Assignment**

0: TCK/SWCLK function selected.

1: PB7 function selected.

- **SYSIO12: PB12 or ERASE Assignment**

0: ERASE function selected.

1: PB12 function selected.

- **CAN1DMABA: CAN1 DMA Base Address**

Give the 16-bit MSB of the CAN1 DMA base address. The 16-bit LSB must be programmed into CAN1 User interface.

Default address is 0x20400000.

Downloaded from Arrow.com.

18.4.8 SMC NAND Flash Chip Select Configuration Register

Name: CCFG_SMCNFCS

Address: 0x40088124

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	SDRAMEN	SMC_NFCS3	SMC_NFCS2	SMC_NFCS1	SMC_NFCS0

- **SMC_NFCS0: SMC NAND Flash Chip Select 0 Assignment**

0: NCS0 is not assigned to a NAND Flash (NANDOE and NANWE not used for NCS0).

1: NCS0 is assigned to a NAND Flash (NANDOE and NANWE used for NCS0).

- **SMC_NFCS1: SMC NAND Flash Chip Select 1 Assignment**

0: NCS1 is not assigned to a NAND Flash (NANDOE and NANWE not used for NCS1).

1: NCS1 is assigned to a NAND Flash (NANDOE and NANWE used for NCS1).

WARNING: This must not be used if SDRAMEN is set.

- **SMC_NFCS2: SMC NAND Flash Chip Select 2 Assignment**

0: NCS2 is not assigned to a NAND Flash (NANDOE and NANWE not used for NCS2).

1: NCS2 is assigned to a NAND Flash (NANDOE and NANWE used for NCS2).

- **SMC_NFCS3: SMC NAND Flash Chip Select 3 Assignment**

0: NCS3 is not assigned to a NAND Flash (NANDOE and NANWE not used for NCS3).

1: NCS3 is assigned to a NAND Flash (NANDOE and NANWE used for NCS3).

- **SDRAMEN: SDRAM Enable**

0: NCS1 is not assigned to SDRAM.

1: NCS1 is assigned to SDRAM.

WARNING: This must not be used if SMC_NFCS1 is set.

Downloaded from Arrow.com.

18.4.9 Write Protection Mode Register

Name: MATRIX_WPMR

Address: 0x400881E4

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							

23	22	21	20	19	18	17	16
WPKEY							

15	14	13	12	11	10	9	8
WPKEY							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x4D4154 (“MAT” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x4D4154 (“MAT” in ASCII).

See Section 18.3.6 ”Register Write Protection” for the list of registers that can be write-protected.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x4D4154	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

Downloaded from Arrow.com.

18.4.10 Write Protection Status Register

Name: MATRIX_WPSR

Address: 0x400881E8

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

- **WPVS: Write Protection Violation Status**

0: No write protection violation has occurred since the last write of the MATRIX_WPMR.

1: A write protection violation has occurred since the last write of the MATRIX_WPMR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protection Violation Source**

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

Downloaded from Arrow.com.

19. USB Transmitter Macrocell Interface (UTMI)

19.1 Description

The USB Transmitter Macrocell Interface (UTMI) registers manage specific aspects of the integrated USB transmitter macrocell functionality not controlled in USB sections.

19.2 Embedded Characteristics

- 32-bit UTMI Registers Control Product-specific Behavior

Downloaded from Arrow.com.

19.3 USB Transmitter Macrocell Interface (UTMI) User Interface

Table 19-1. Register Mapping

Offset	Register	Name	Access	Reset
0x00-0x0C	Reserved	–	–	–
0x10	OHCI Interrupt Configuration Register	UTMI_OHCIICR	Read/Write	0x0
0x14–0x2C	Reserved	–	–	–
0x30	UTMI Clock Trimming Register	UTMI_CKTRIM	Read/Write	0x00010000
0x34–0x3C	Reserved	–	–	–
0x40-0xFC	Reserved	–	–	–

Downloaded from Arrow.com.

19.3.1 OHCI Interrupt Configuration Register

Name: UTMI_OHCIICR

Address: 0x400E0410

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
UDPPUDIS	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	APPSTART	ARIE	–	–	–	RES0

- **RESx: USB PORTx Reset**

0: Resets USB port.

1: Usable USB port.

- **ARIE: OHCI Asynchronous Resume Interrupt Enable**

0: Interrupt disabled.

1: Interrupt enabled.

- **APPSTART: Reserved**

0: Must write 0.

- **UDPPUDIS: USB Device Pull-up Disable**

0: USB device pull-up connection is enabled.

1: USB device pull-up connection is disabled.

Downloaded from Arrow.com.

19.3.2 UTMI Clock Trimming Register

Name: UTMI_CKTRIM

Address: 0x400E0430

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	FREQ	

- **FREQ: UTMI Reference Clock Frequency**

Value	Name	Description
0	XTAL12	12 MHz reference clock
1	XTAL16	16 MHz reference clock

Downloaded from Arrow.com.

20. Chip Identifier (CHIPID)

20.1 Description

Chip Identifier (CHIPID) registers are used to recognize the device and its revision. These registers provide the sizes and types of the on-chip memories, as well as the set of embedded peripherals.

Two CHIPID registers are embedded: Chip ID Register (CHIPID_CIDR) and Chip ID Extension Register (CHIPID_EXID). Both registers contain a hard-wired value that is read-only.

The CHIPID_CIDR register contains the following fields:

- VERSION: Identifies the revision of the silicon
- EPROC: Indicates the embedded ARM processor
- NVPTYP and NVPSIZ: Identify the type of embedded non-volatile memory and the size
- SRAMSIZ: Indicates the size of the embedded SRAM
- ARCH: Identifies the set of embedded peripherals
- EXT: Shows the use of the extension identifier register

The CHIPID_EXID register is device-dependent and reads 0 if CHIPID_CIDR.EXT = 0.

20.2 Embedded Characteristics

- Chip ID Registers
  - Identification of the Device Revision, Sizes of the Embedded Memories, Set of Peripherals, Embedded Processor

Table 20-1. Chip ID Registers

Chip Name	CHIPID_CIDR	CHIPID_EXID
SAME70Q21	0xA102_0E00	0x00000002
SAME70Q20	0xA102_0C00	0x00000002
SAME70Q19	0xA10D_0A00	0x00000002
SAME70N21	0xA102_0E00	0x00000001
SAME70N20	0xA102_0C00	0x00000001
SAME70N19	0xA10D_0A00	0x00000001
SAME70J21	0xA102_0E00	0x00000000
SAME70J20	0xA102_0C00	0x00000000
SAME70J19	0xA10D_0A00	0x00000000
SAMS70Q21	0xA112_0E00	0x00000002
SAMS70Q20	0xA112_0C00	0x00000002
SAMS70Q19	0xA11D_0A00	0x00000002
SAMS70N21	0xA112_0E00	0x00000001
SAMS70N20	0xA112_0C00	0x00000001
SAMS70N19	0xA11D_0A00	0x00000001
SAMS70J21	0xA1120_E00	0x00000000
SAMS70J20	0xA112_0C00	0x00000000
SAMS70J19	0xA11D_0A00	0x00000000
SAMV71Q21	0xA122_0E00	0x00000002
SAMV71Q20	0xA122_0C00	0x00000002

Downloaded from Arrow.com.

Table 20-1. Chip ID Registers (Continued)

Chip Name	CHIPID_CIDR	CHIPID_EXID
SAMV71Q19	0xA12D_0A00	0x00000002
SAMV71N21	0xA122_0E00	0x00000001
SAMV71N20	0xA122_0C00	0x00000001
SAMV71N19	0xA12D_0A00	0x00000001
SAMV71J21	0xA122_0E00	0x00000000
SAMV71J20	0xA122_0C00	0x00000000
SAMV71J19	0xA12D_0A00	0x00000000
SAMV70Q20	0xA132_0C00	0x00000002
SAMV70Q19	0xA13D_0A00	0x00000002
SAMV70N20	0xA132_0C00	0x00000001
SAMV70N19	0xA13D_0A00	0x00000001
SAMV70J20	0xA1320_C00	0x00000000
SAMV70J19	0xA13D_0A00	0x00000000

Downloaded from Arrow.com.

20.3 Chip Identifier (CHIPID) User Interface

Table 20-2. Register Mapping

Offset	Register	Name	Access	Reset
0x0	Chip ID Register	CHIPID_CIDR	Read-only	–
0x4	Chip ID Extension Register	CHIPID_EXID	Read-only	–

Downloaded from Arrow.com.

20.3.1 Chip ID Register

Name: CHIPID_CIDR

Address: 0x400E0940

Access: Read-only

31	30	29	28	27	26	25	24
EXT	NVPTYP			ARCH			

23	22	21	20	19	18	17	16
ARCH				SRAMSIZ			

15	14	13	12	11	10	9	8
NVPSIZ2				NVPSIZ			

7	6	5	4	3	2	1	0
EPROC			VERSION				

- **VERSION: Version of the Device**

Current version of the device.

- **EPROC: Embedded Processor**

Value	Name	Description
0	SAM x7	Cortex-M7
1	ARM946ES	ARM946ES
2	ARM7TDMI	ARM7TDMI
3	CM3	Cortex-M3
4	ARM920T	ARM920T
5	ARM926EJS	ARM926EJS
6	CA5	Cortex-A5
7	CM4	Cortex-M4

- **NVPSIZ: Nonvolatile Program Memory Size**

Value	Name	Description
0	NONE	None
1	8K	8 Kbytes
2	16K	16 Kbytes
3	32K	32 Kbytes
4	–	Reserved
5	64K	64 Kbytes
6	–	Reserved
7	128K	128 Kbytes
8	160K	160 Kbytes
9	256K	256 Kbytes
10	512K	512 Kbytes

Downloaded from Arrow.com.

Value	Name	Description
11	–	Reserved
12	1024K	1024 Kbytes
13	–	Reserved
14	2048K	2048 Kbytes
15	–	Reserved

- **NVPSIZ2: Second Nonvolatile Program Memory Size**

Value	Name	Description
0	NONE	None
1	8K	8 Kbytes
2	16K	16 Kbytes
3	32K	32 Kbytes
4	–	Reserved
5	64K	64 Kbytes
6	–	Reserved
7	128K	128 Kbytes
8	–	Reserved
9	256K	256 Kbytes
10	512K	512 Kbytes
11	–	Reserved
12	1024K	1024 Kbytes
13	–	Reserved
14	2048K	2048 Kbytes
15	–	Reserved

- **SRAMSIZ: Internal SRAM Size**

Value	Name	Description
0	48K	48 Kbytes
1	192K	192 Kbytes
2	384K	384 Kbytes
3	6K	6 Kbytes
4	24K	24 Kbytes
5	4K	4 Kbytes
6	80K	80 Kbytes
7	160K	160 Kbytes
8	8K	8 Kbytes
9	16K	16 Kbytes
10	32K	32 Kbytes
11	64K	64 Kbytes

Downloaded from Arrow.com.

Value	Name	Description
12	128K	128 Kbytes
13	256K	256 Kbytes
14	96K	96 Kbytes
15	512K	512 Kbytes

- **ARCH: Architecture Identifier**

Value	Name	Description
0x10	SAM E70	SAM E70
0x11	SAM S70	SAM S70
0x12	SAM V71	SAM V71
0x13	SAM V70	SAM V70

- **NVPTYP: Nonvolatile Program Memory Type**

Value	Name	Description
0	ROM	ROM
1	ROMLESS	ROMless or on-chip Flash
2	FLASH	Embedded Flash Memory
3	ROM_FLASH	ROM and Embedded Flash Memory • NVPSIZ is ROM size • NVPSIZ2 is Flash size
4	SRAM	SRAM emulating ROM

- **EXT: Extension Flag**

0: Chip ID has a single register definition without extension.

1: An extended Chip ID exists.

Downloaded from Arrow.com.

20.3.2 Chip ID Extension Register

Name: CHIPID_EXID

Address: 0x400E0944

Access: Read-only

31	30	29	28	27	26	25	24
EXID							

23	22	21	20	19	18	17	16
EXID							

15	14	13	12	11	10	9	8
EXID							

7	6	5	4	3	2	1	0
EXID							

- **EXID: Chip ID Extension**

This field is cleared if CHIPID_CIDR.EXT = 0.

Downloaded from Arrow.com.

21. Enhanced Embedded Flash Controller (EEFC)

21.1 Description

The Enhanced Embedded Flash Controller (EEFC) provides the interface of the Flash block with the 32-bit internal bus.

Its 128-bit wide memory interface increases performance. It also manages the programming, erasing, locking and unlocking sequences of the Flash using a full set of commands. One of the commands returns the embedded Flash descriptor definition that informs the system about the Flash organization, thus making the software generic.

21.2 Embedded Characteristics

- Increases Performance in Thumb-2 Mode with 128-bit-wide Memory Interface up to 150 MHz
- Code Loop Optimization
- 128 Lock Bits, Each Protecting a Lock Region
- 9 General-purpose GPNVM Bits
- One-by-one Lock Bit Programming
- Commands Protected by a Keyword
- Erase the Entire Flash
- Erase by Plane
- Erase by Sector
- Erase by Page
- Provides Unique Identifier
- Provides 512-byte User Signature Area
- Supports Erasing before Programming
- Locking and Unlocking Operations
- ECC Single and Multiple Error Flags Report
- Supports Read of the Calibration Bits
- Register Write Protection

21.3 Product Dependencies

21.3.1 Power Management

The Enhanced Embedded Flash Controller (EEFC) is continuously clocked. The Power Management Controller has no effect on its behavior.

21.3.2 Interrupt Sources

The EEFC interrupt line is connected to the interrupt controller. Using the EEFC interrupt requires the interrupt controller to be programmed first. The EEFC interrupt is generated only if the value of bit EEFC_FMR.FRDY is 1.

Table 21-1. Peripheral IDs

Instance	ID
EFC	6

Downloaded from Arrow.com.

21.4 Functional Description

21.4.1 Embedded Flash Organization

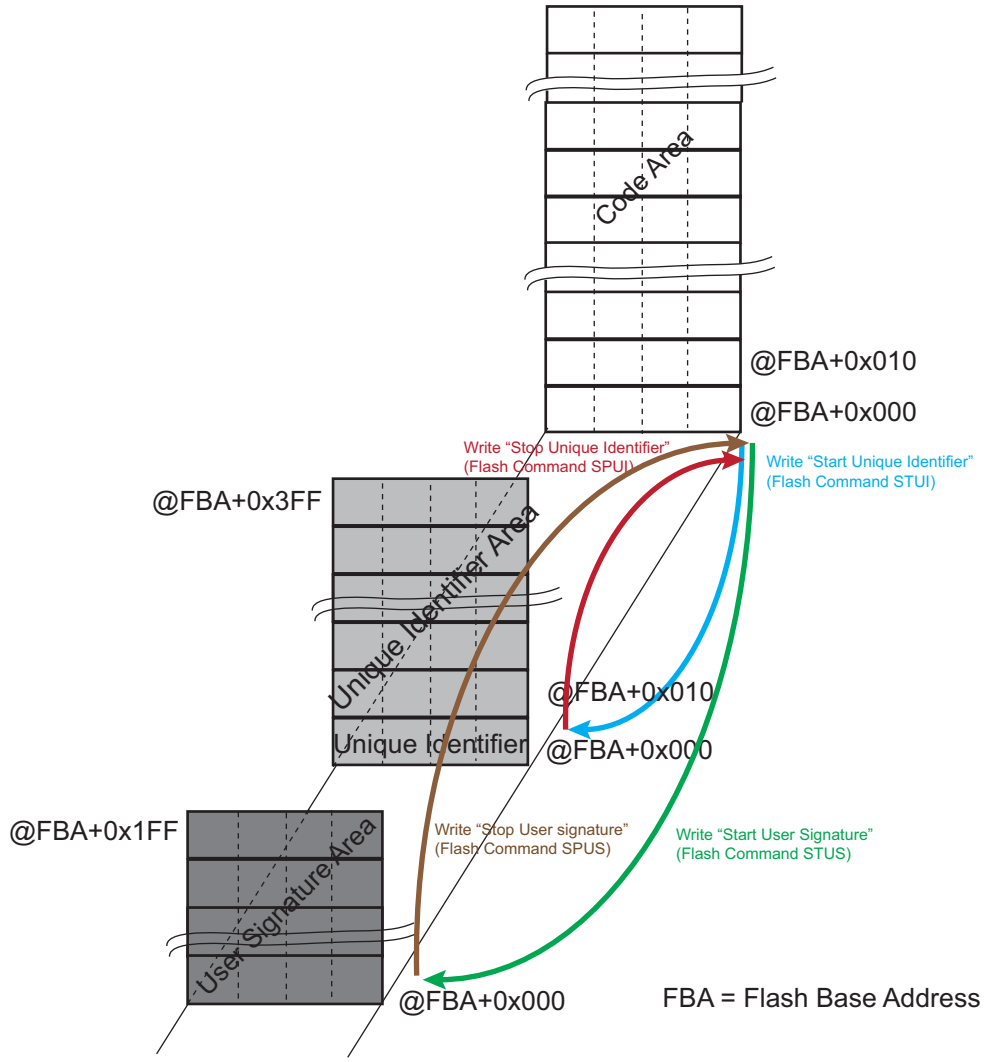
The embedded Flash interfaces directly with the internal bus. The embedded Flash is composed of:

- One memory plane organized in several pages of the same size for the code
- A separate 2 x 512-byte memory area which includes the unique chip identifier
- A separate 512-byte memory area for the user signature
- Two 128-bit read buffers used for code read optimization
- One 128-bit read buffer used for data read optimization
- One write buffer that manages page programming. The write buffer size is equal to the page size. This buffer is write-only and accessible all along the 1 Mbyte address space, so that each word can be written to its final address.
- Several lock bits used to protect write/erase operation on several pages (lock region). A lock bit is associated with a lock region composed of several pages in the memory plane.
- Several bits that may be set and cleared through the EEFC interface, called general-purpose non-volatile memory bits (GPNVM bits)

The embedded Flash size, the page size, the organization of lock regions and the definition of GPNVM bits are specific to the device. The EEFC returns a descriptor of the Flash controller after a ‘Get Flash Descriptor’ command has been issued by the application (see Section 21.4.3.1 ”Get Flash Descriptor Command”).

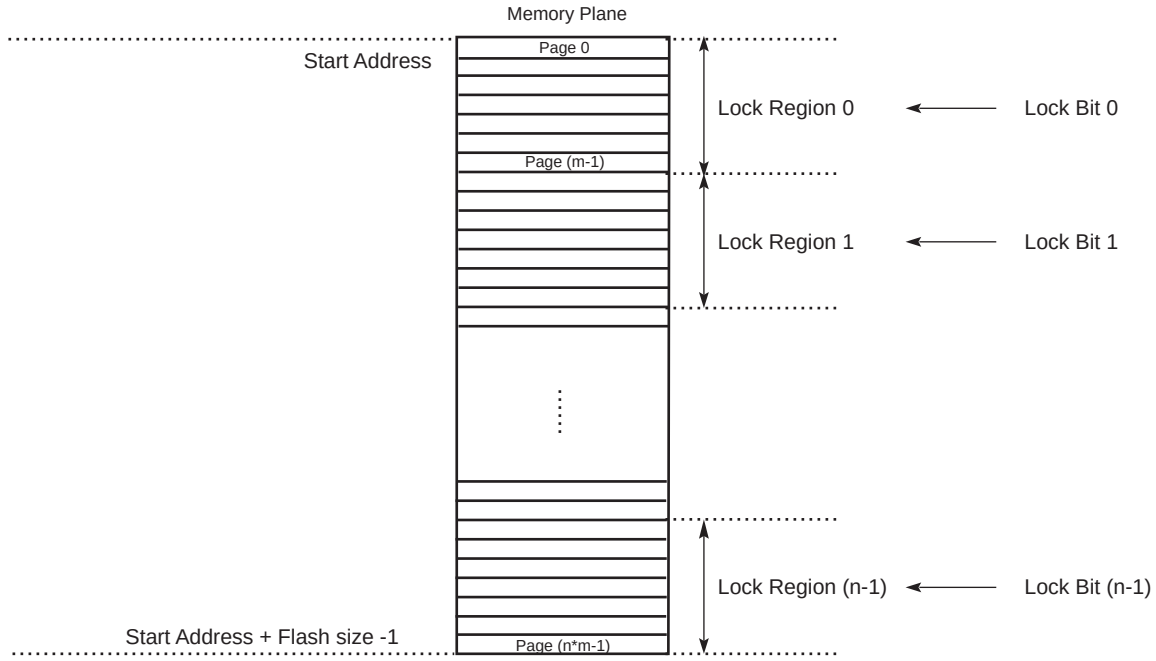
Downloaded from Arrow.com.

Figure 21-1. Flash Memory Areas



Downloaded from Arrow.com.

Figure 21-2. Organization of Embedded Flash for Code



Downloaded from Arrow.com.

21.4.2 Read Operations

An optimized controller manages embedded Flash reads, thus increasing performance when the processor is running in Thumb-2 mode by means of the 128-bit-wide memory interface.

The Flash memory is accessible through 8-, 16- and 32-bit reads.

As the Flash block size is smaller than the address space reserved for the internal memory area, the embedded Flash wraps around the address space and appears to be repeated within it.

The read operations can be performed with or without wait states. Wait states must be programmed in the field FWS in the Flash Mode register (EEFC_FMR). Defining FWS as 0 enables the single-cycle access of the embedded Flash. For more details, refer to the section Section 56. "Electrical Characteristics" of this datasheet.

21.4.2.1 Code Read Optimization

Code read optimization is enabled if the bit EEFC_FMR.SCOD is cleared.

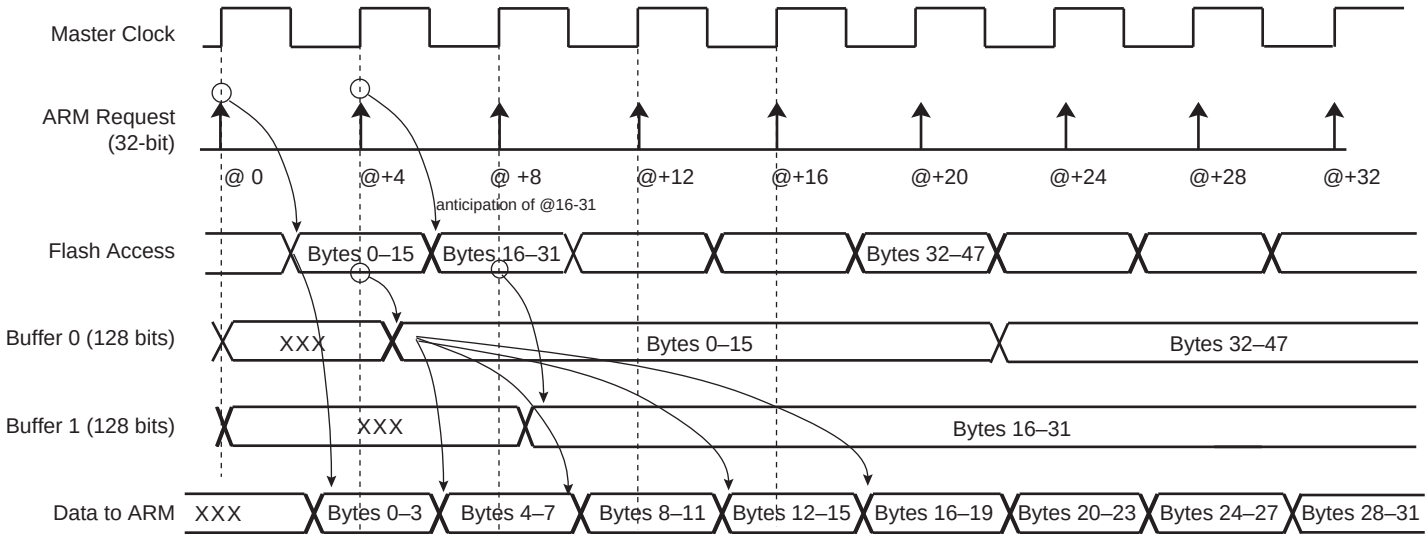
A system of 2 x 128-bit buffers is added in order to optimize sequential code fetch.

Note: Immediate consecutive code read accesses are not mandatory to benefit from this optimization.

The sequential code read optimization is enabled by default. If the bit EEFC_FMR.SCOD is set to 1, these buffers are disabled and the sequential code read is no longer optimized.

Another system of 2 x 128-bit buffers is added in order to optimize loop code fetch. Refer to Section 21.4.2.2 "Code Loop Optimization" for more details.

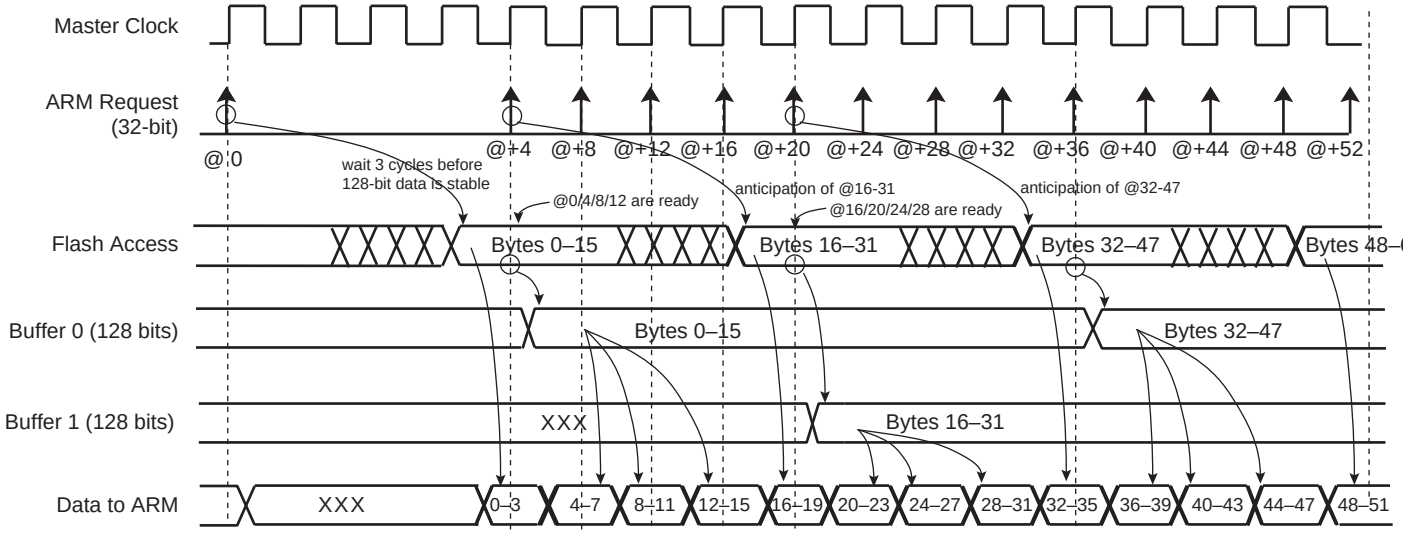
Figure 21-3. Code Read Optimization for FWS = 0



Note: When FWS is equal to 0, all the accesses are performed in a single-cycle access.

Downloaded from Arrow.com.

Figure 21-4. Code Read Optimization for FWS = 3



Note: When FWS is between 1 and 3, in case of sequential reads, the first access takes (FWS + 1) cycles. The following accesses take only one cycle.

21.4.2.2 Code Loop Optimization

Code loop optimization is enabled when the bit EEFC_FMR.CLOE is set to 1.

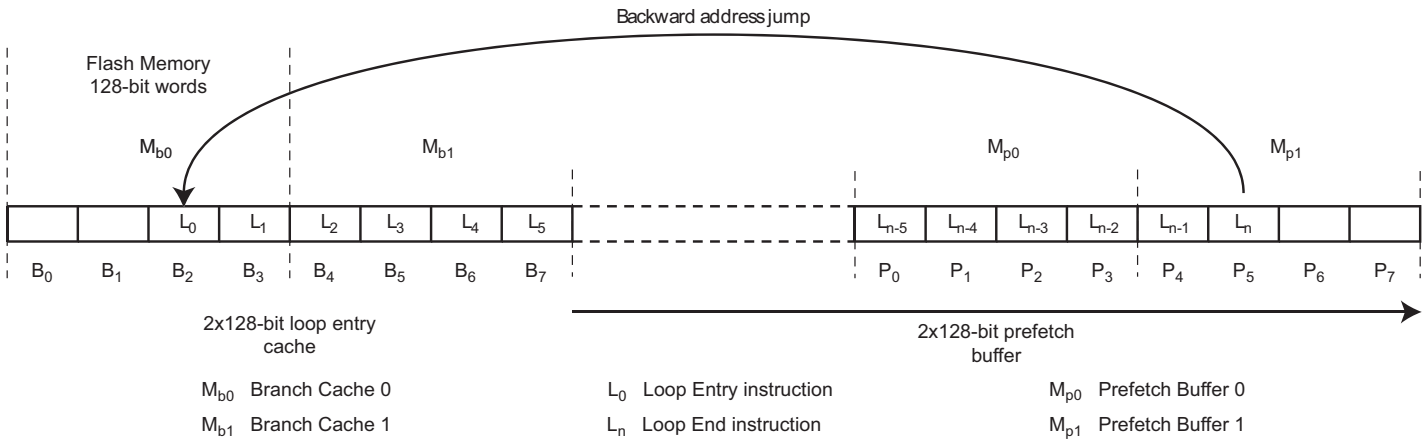
When a backward jump is inserted in the code, the pipeline of the sequential optimization is broken and becomes inefficient. In this case, the loop code read optimization takes over from the sequential code read optimization to prevent the insertion of wait states. The loop code read optimization is enabled by default. In EEFC_FMR, if the bit CLOE is reset to 0 or the bit SCOD is set to 1, these buffers are disabled and the loop code read is not optimized.

When code loop optimization is enabled, if inner loop body instructions L_0 to L_n are positioned from the 128-bit Flash memory cell M_{b0} to the memory cell M_{p1}, after recognition of a first backward branch, the first two Flash memory cells M_{b0} and M_{b1} targeted by this branch are cached for fast access from the processor at the next loop iteration.

Then by combining the sequential prefetch (described in Section 21.4.2.1 ”Code Read Optimization”) through the loop body with the fast read access to the loop entry cache, the entire loop can be iterated with no wait state.

Figure 21-5 illustrates code loop optimization.

Figure 21-5. Code Loop Optimization



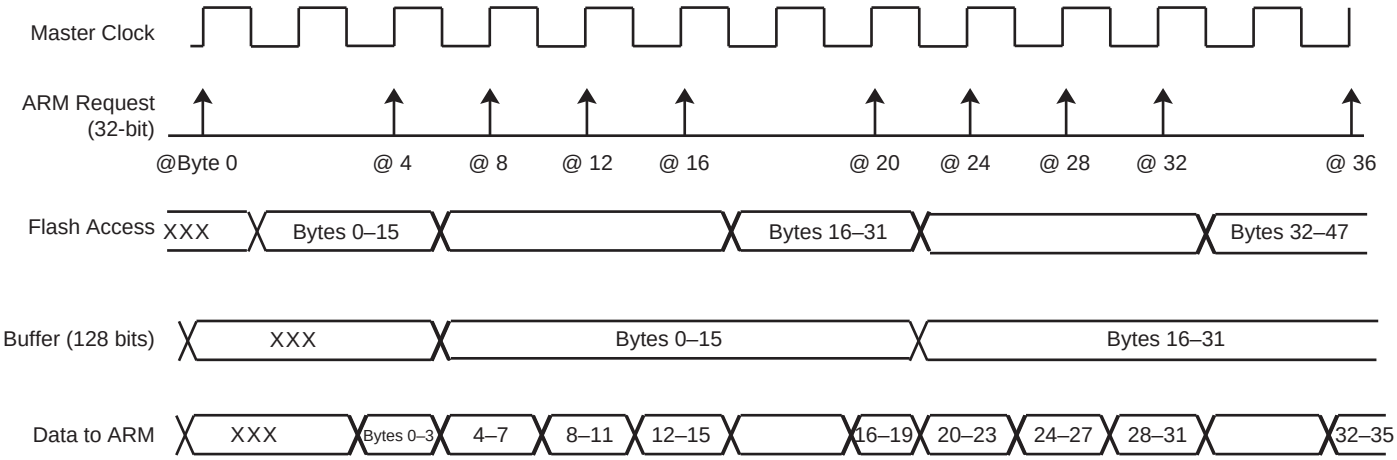
Downloaded from Arrow.com.

21.4.2.3 Data Read Optimization

The organization of the Flash in 128 bits is associated with two 128-bit prefetch buffers and one 128-bit data read buffer, thus providing maximum system performance. This buffer is added in order to store the requested data plus all the data contained in the 128-bit aligned data. This speeds up sequential data reads if, for example, FWS is equal to 1 (see Figure 21-6). The data read optimization is enabled by default. If the bit EEFC_FMR.SCOD is set to 1, this buffer is disabled and the data read is no longer optimized.

Note: No consecutive data read accesses are mandatory to benefit from this optimization.

Figure 21-6. Data Read Optimization for FWS = 1



Downloaded from Arrow.com.

21.4.3 Flash Commands

The EEFC offers a set of commands to manage programming the Flash memory, locking and unlocking lock regions, consecutive programming, locking and full Flash erasing, etc.

The commands are listed in the following table.

Table 21-2. Set of Commands

Command	Value	Mnemonic
Get Flash descriptor	0x00	GETD
Write page	0x01	WP
Write page and lock	0x02	WPL
Erase page and write page	0x03	EWP
Erase page and write page then lock	0x04	EWPL
Erase all	0x05	EA
Erase pages	0x07	EPA
Set lock bit	0x08	SLB
Clear lock bit	0x09	CLB
Get lock bit	0x0A	GLB
Set GPNVM bit	0x0B	SGPB
Clear GPNVM bit	0x0C	CGPB
Get GPNVM bit	0x0D	GGPB
Start read unique identifier	0x0E	STUI
Stop read unique identifier	0x0F	SPUI
Get CALIB bit	0x10	GCALB
Erase sector	0x11	ES
Write user signature	0x12	WUS
Erase user signature	0x13	EUS
Start read user signature	0x14	STUS
Stop read user signature	0x15	SPUS

In order to execute one of these commands, select the required command using the FCMD field in the Flash Command register (EEFC_FCR). As soon as EEFC_FCR is written, the FRDY flag and the FVALUE field in the Flash Result register (EEFC_FRR) are automatically cleared. Once the current command has completed, the FRDY flag is automatically set. If an interrupt has been enabled by setting the bit EEFC_FMR.FRDY, the corresponding interrupt line of the interrupt controller is activated. (Note that this is true for all commands except for the STUI command. The FRDY flag is not set when the STUI command has completed.)

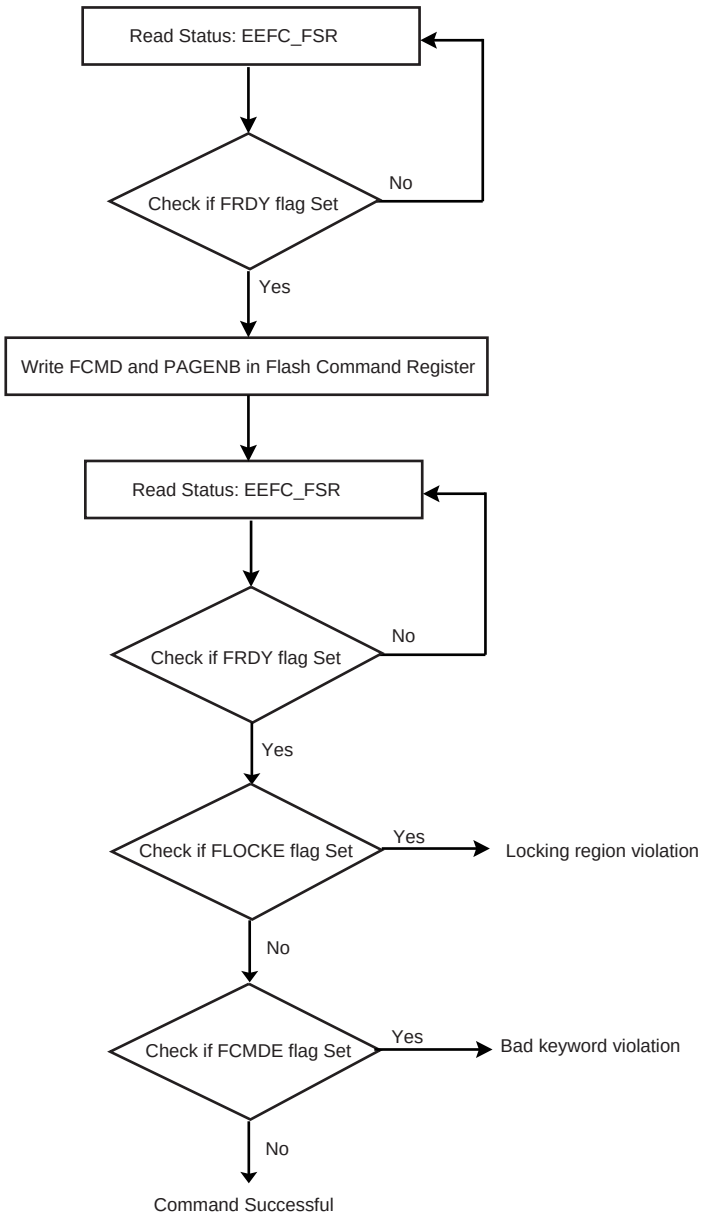
All the commands are protected by the same keyword, which must be written in the eight highest bits of EEFC_FCR.

Writing EEFC_FCR with data that does not contain the correct key and/or with an invalid command has no effect on the whole memory plane, but the FCMDE flag is set in the Flash Status register (EEFC_FSR). This flag is automatically cleared by a read access to EEFC_FSR.

When the current command writes or erases a page in a locked region, the command has no effect on the whole memory plane, but the FLOCKE flag is set in EEFC_FSR. This flag is automatically cleared by a read access to EEFC_FSR.

Downloaded from Arrow.com.

Figure 21-7. Command State Chart



Downloaded from Arrow.com.

21.4.3.1 Get Flash Descriptor Command

This command provides the system with information on the Flash organization. The system can take full advantage of this information. For instance, a device could be replaced by one with more Flash capacity, and so the software is able to adapt itself to the new configuration.

To get the embedded Flash descriptor, the application writes the GETD command in EEFC_FCR. The first word of the descriptor can be read by the software application in EEFC_FRR as soon as the FRDY flag in EEFC_FSR rises. The next reads of EEFC_FRR provide the following word of the descriptor. If extra read operations to EEFC_FRR are done after the last word of the descriptor has been returned, the EEFC_FRR value is 0 until the next valid command.

Table 21-3. Flash Descriptor Definition

Symbol	Word Index	Description
FL_ID	0	Flash interface description
FL_SIZE	1	Flash size in bytes
FL_PAGE_SIZE	2	Page size in bytes
FL_NB_PLANE	3	Number of planes
FL_PLANE[0]	4	Number of bytes in the plane
FL_NB_LOCK	4 + FL_NB_PLANE	Number of lock bits. A bit is associated with a lock region. A lock bit is used to prevent write or erase operations in the lock region.
FL_LOCK[0]	4 + FL_NB_PLANE + 1	Number of bytes in the first lock region

21.4.3.2 Write Commands

DMA write accesses must be 32-bit aligned. If a single byte is to be written in a 32-bit word, the rest of the word must be written with ones.

Several commands are used to program the Flash.

Only 0 values can be programmed using Flash technology; 1 is the erased value. In order to program words in a page, the page must first be erased. Commands are available to erase the full memory plane or a given number of pages. With the EWP and EWPL commands, a page erase is done automatically before a page programming.

After programming, the page (the entire lock region) can be locked to prevent miscellaneous write or erase sequences. The lock bit can be automatically set after page programming using WPL or EWPL commands.

Data to be programmed in the Flash must be written in an internal latch buffer before writing the programming command in EEFC_FCR. Data can be written at their final destination address, as the latch buffer is mapped into the Flash memory address space and wraps around within this Flash address space.

Byte and half-word AHB accesses to the latch buffer are not allowed. Only 32-bit word accesses are supported.

32-bit words must be written continuously, in either ascending or descending order. Writing the latch buffer in a random order is not permitted. This prevents mapping a C-code structure to the latch buffer and accessing the data of the structure in any order. It is instead recommended to fill in a C-code structure in SRAM and copy it in the latch buffer in a continuous order.

Write operations in the latch buffer are performed with the number of wait states programmed for reading the Flash.

The latch buffer is automatically re-initialized, i.e., written with logical 1, after execution of each programming command.

The programming sequence is the following:

Downloaded from Arrow.com.

1. Write the data to be programmed in the latch buffer.
2. Write the programming command in EEFC_FCR. This automatically clears the bit EEFC_FSR.FRDY.
3. When Flash programming is completed, the bit EEFC_FSR.FRDY rises. If an interrupt has been enabled by setting the bit EEFC_FMR.FRDY, the interrupt line of the EEFC is activated.

Three errors can be detected in EEFC_FSR after a programming sequence:

- Command Error: A bad keyword has been written in EEFC_FCR.
- Lock Error: The page to be programmed belongs to a locked region. A command must be run previously to unlock the corresponding region.
- Flash Error: When programming is completed, the WriteVerify test of the Flash memory has failed.

Only one page can be programmed at a time. It is possible to program all the bits of a page (full page programming) or only some of the bits of the page (partial page programming).

Depending on the number of bits to be programmed within the page, the EEFC adapts the write operations required to program the Flash.

When a 'Write Page' (WP) command is issued, the EEFC starts the programming sequence and all the bits written at 0 in the latch buffer are cleared in the Flash memory array.

During programming, i.e., until EEFC_FSR.FDRY rises, access to the Flash is not allowed.

Full Page Programming

To program a full page, all the bits of the page must be erased before writing the latch buffer and issuing the WP command. The latch buffer must be written in ascending order, starting from the first address of the page. See Figure 21-8 "Full Page Programming".

Partial Page Programming

To program only part of a page using the WP command, the following constraints must be respected:

- Data to be programmed must be contained in integer multiples of 128-bit address-aligned words.
- 128-bit words can be programmed only if all the corresponding bits in the Flash array are erased (at logical value 1).

See Figure 21-9 "Partial Page Programming".

Optimized Partial Page Programming

The EEFC automatically detects the number of 128-bit words to be programmed. If only one 128-bit aligned word is to be programmed in the Flash array, the process is optimized to reduce the time needed for programming.

If several 128-bit words are to be programmed, a standard page programming operation is performed.

See Figure 21-10 "Optimized Partial Page Programming".

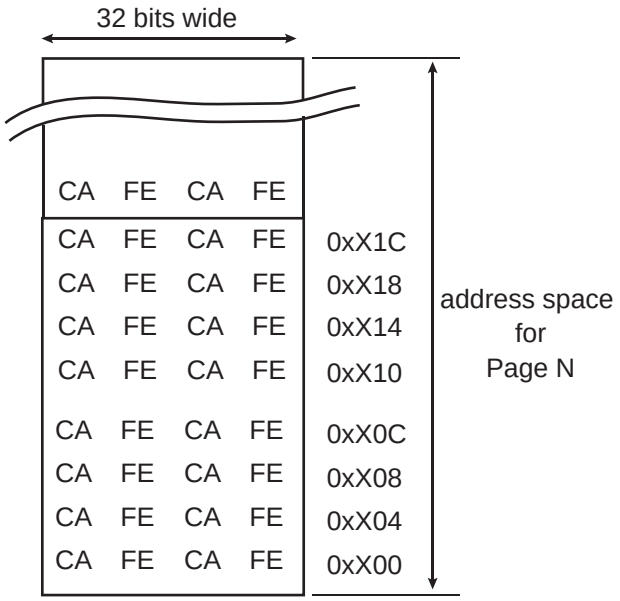
Programming Bytes

Individual bytes can be programmed using the Partial page programming mode.
In this case, an area of 128 bits must be reserved for each byte.

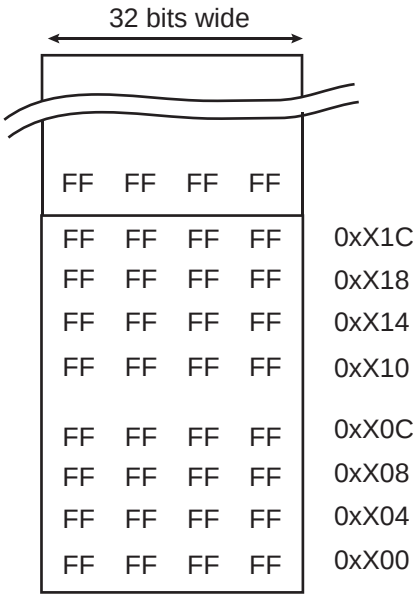
Refer to Figure 21-11 "Programming Bytes in the Flash".

Downloaded from Arrow.com.

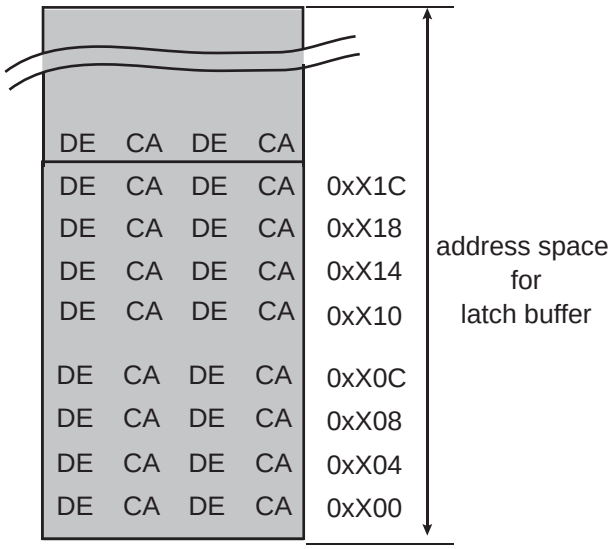
Figure 21-8. Full Page Programming



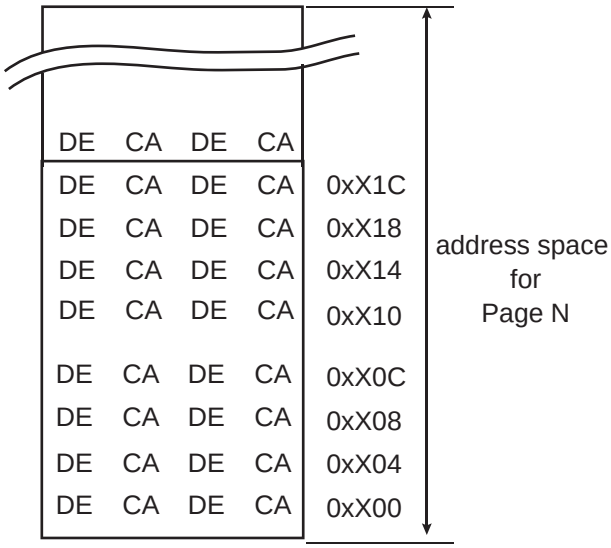
Before programming: Unerased page in Flash array



Step 1: Flash array after page erase



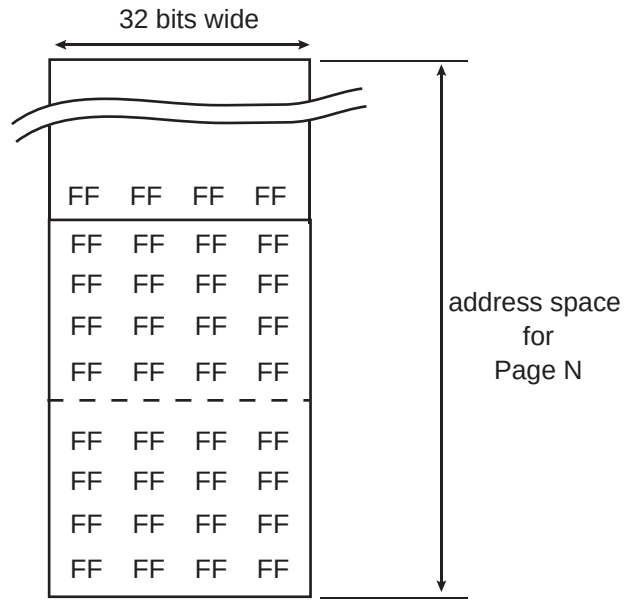
Step 2: Writing a page in the latch buffer



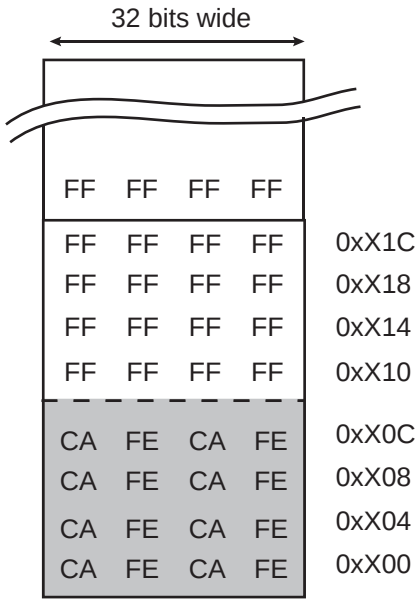
Step 3: Page in Flash array after issuing WP command and FRDY=1

Downloaded from Arrow.com.

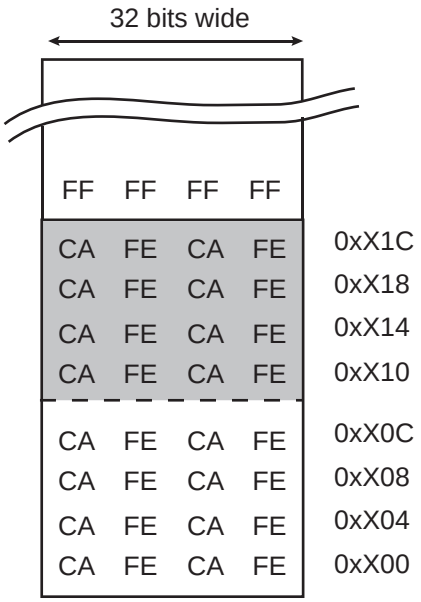
Figure 21-9. Partial Page Programming



Step 1: Flash array after page erase



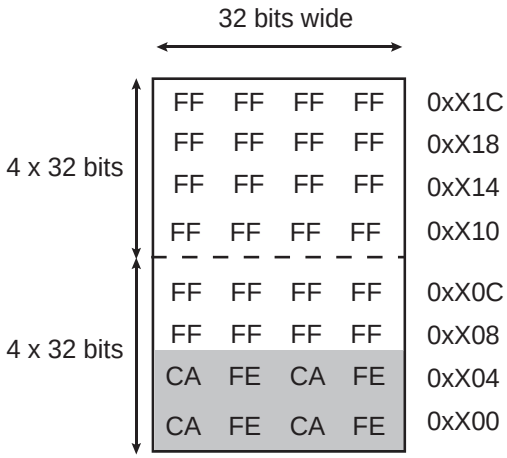
Step 2: Flash array after programming
128-bit at address 0xX00 (write latch buffer + WP)



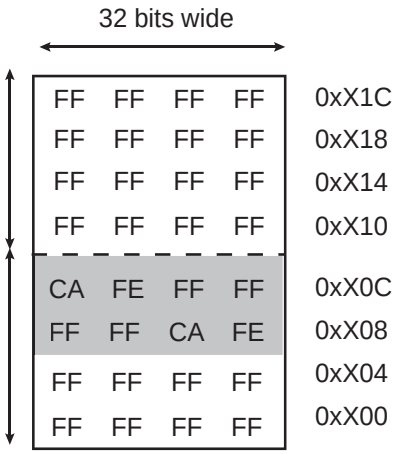
Step 3: Flash array after programming
a second 128-bit data at address 0xX10
(write latch buffer + WP)

Downloaded from Arrow.com.

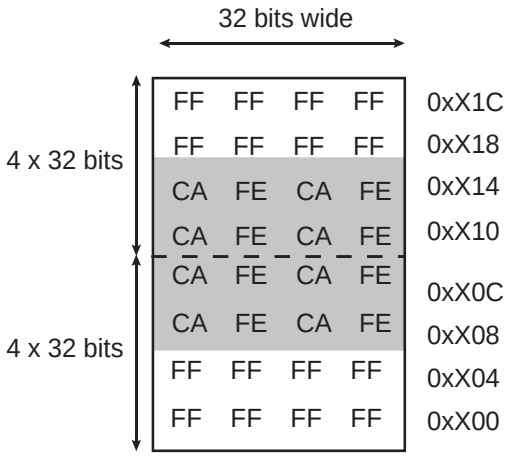
Figure 21-10. Optimized Partial Page Programming



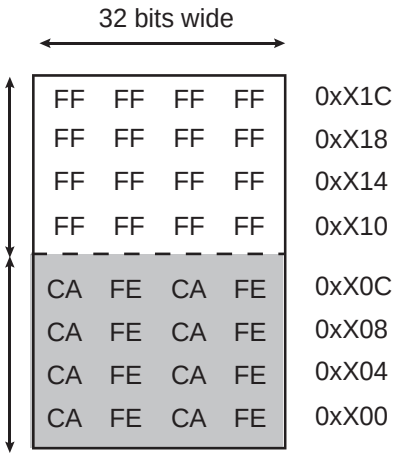
Case 1: 2 x 32 bits modified, not crossing 128-bit boundary
User programs WP, Flash Controller sends Write Word
=> Only 1 word programmed => programming period reduced



Case 2: 2 x 32 bits modified, not crossing 128-bit boundary
User programs WP, Flash Controller sends Write Word
=> Only 1 word programmed => programming period reduced



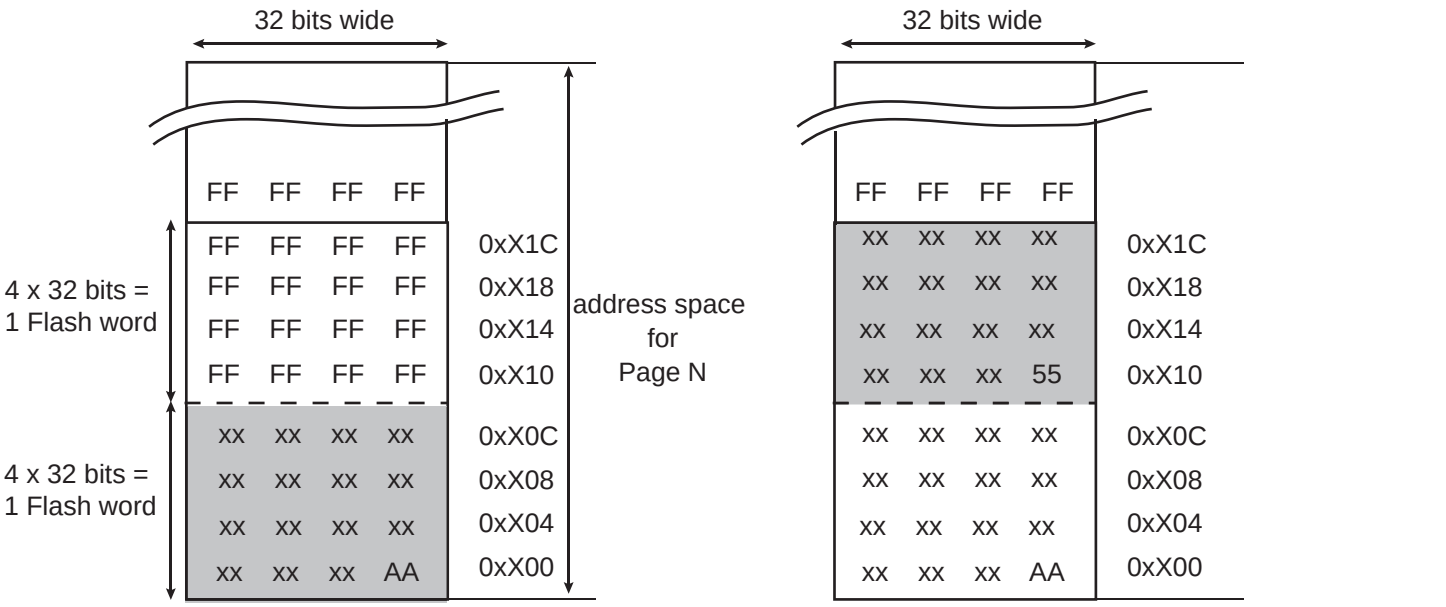
Case 3: 4 x 32 bits modified across 128-bit boundary
User programs WP, Flash Controller sends WP
=> Whole page programmed



Case 4: 4 x 32 bits modified, not crossing 128-bit boundary
User programs WP, Flash Controller sends Write Word
=> Only 1 word programmed => programming period reduced

Downloaded from Arrow.com.

Figure 21-11. Programming Bytes in the Flash



Step 1: Flash array after programming first byte (0xAA)
128-bit used at address 0xX00 (write latch buffer + WP)

Step 2: Flash array after programming second byte (0x55)
128-bit used at address 0xX10 (write latch buffer + WP)

Note: The byte location shown here is for example only, it can be any byte location within a 64-bit word

21.4.3.3 Erase Commands

Erase commands are allowed only on unlocked regions. Depending on the Flash memory, several commands can be used to erase the Flash:

- Erase All Memory (EA): All memory is erased. The processor must not fetch code from the Flash memory.
- Erase Pages (EPA): 8 or 16 pages are erased in the Flash sector selected. The first page to be erased is specified in the FARG[15:2] field of the EEFC_FCR. The first page number must be a multiple of 8, 16 or 32 depending on the number of pages to erase at the same time.
- Erase Sector (ES): A full memory sector is erased. Sector size depends on the Flash memory. EEFC_FCR.FARG must be set with a page number that is in the sector to be erased.

Note: If one subsector is locked within the first sector, the Erase Sector (ES) command cannot be processed on non-locked subsectors of the first sector. All the lock bits of the first sector must be cleared prior to issuing an ES command on the first sector. After the ES command has been issued, the first sector lock bits must be reverted to the state before clearing them.

If the processor is fetching code from the Flash memory while the EPA or ES command is being executed, the processor accesses are stalled until the EPA command is completed. To avoid stalling the processor, the code can be run out of internal SRAM.

The erase sequence is the following:

Downloaded from Arrow.com.

1. Erase starts as soon as one of the erase commands and the FARG field are written in EEFC_FCR.
   – For the EPA command, the two lowest bits of the FARG field define the number of pages to be erased (FARG[1:0]):

Table 21-4. EEFC_FCR.FARG Field for EPA Command

FARG[1:0]	Number of pages to be erased with EPA command
0	4 pages (only valid for small 8 KB sectors)
1	8 pages (only valid for small 8 KB sectors)
2	16 pages
3	32 pages (not valid for small 8 KB sectors)

2. When erasing is completed, the bit EEFC_FSR.FRDY rises. If an interrupt has been enabled by setting the bit EEFC_FMR.FRDY, the interrupt line of the interrupt controller is activated.

Three errors can be detected in EEFC_FSR after an erasing sequence:

- Command Error: A bad keyword has been written in EEFC_FCR.
- Lock Error: At least one page to be erased belongs to a locked region. The erase command has been refused, no page has been erased. A command must be run previously to unlock the corresponding region.
- Flash Error: At the end of the erase period, the EraseVerify test of the Flash memory has failed.

21.4.3.4 Lock Bit Protection

Lock bits are associated with several pages in the embedded Flash memory plane. This defines lock regions in the embedded Flash memory plane. They prevent writing/erasing protected pages.

The lock sequence is the following:

1. Execute the ‘Set Lock Bit’ command by writing EEFC_FCR.FCMD with the SLB command and EEFC_FCR.FARG with a page number to be protected.
2. When the locking completes, the bit EEFC_FSR.FRDY rises. If an interrupt has been enabled by setting the bit EEFC_FMR.FRDY, the interrupt line of the interrupt controller is activated.
3. The result of the SLB command can be checked running a ‘Get Lock Bit’ (GLB) command.

Note: The value of the FARG argument passed together with SLB command must not exceed the higher lock bit index available in the product.

Two errors can be detected in EEFC_FSR after a programming sequence:

- Command Error: A bad keyword has been written in EEFC_FCR.
- Flash Error: At the end of the programming, the EraseVerify or WriteVerify test of the Flash memory has failed.

It is possible to clear lock bits previously set. After the lock bits are cleared, the locked region can be erased or programmed. The unlock sequence is the following:

1. Execute the ‘Clear Lock Bit’ command by writing EEFC_FCR.FCMD with the CLB command and EEFC_FCR.FARG with a page number to be unprotected.
2. When the unlock completes, the bit EEFC_FSR.FRDY rises. If an interrupt has been enabled by setting the bit EEFC_FMR.FRDY, the interrupt line of the interrupt controller is activated.

Note: The value of the FARG argument passed together with CLB command must not exceed the higher lock bit index available in the product.

Two errors can be detected in EEFC_FSR after a programming sequence:

- Command Error: A bad keyword has been written in EEFC_FCR.
- Flash Error: At the end of the programming, the EraseVerify or WriteVerify test of the Flash memory has failed.

Downloaded from Arrow.com.

The status of lock bits can be returned by the EEFC. The ‘Get Lock Bit’ sequence is the following:

1. Execute the ‘Get Lock Bit’ command by writing EEFC_FCR.FCMD with the GLB command. Field EEFC_FCR.FARG is meaningless.
2. Lock bits can be read by the software application in EEFC_FRR. The first word read corresponds to the 32 first lock bits, next reads providing the next 32 lock bits as long as it is meaningful. Extra reads to EEFC_FRR return 0.

For example, if the third bit of the first word read in EEFC_FRR is set, the third lock region is locked.

Two errors can be detected in EEFC_FSR after a programming sequence:

- Command Error: A bad keyword has been written in EEFC_FCR.
- Flash Error: At the end of the programming, the EraseVerify or WriteVerify test of the Flash memory has failed.

Note: Access to the Flash in read is permitted when a ‘Set Lock Bit’, ‘Clear Lock Bit’ or ‘Get Lock Bit’ command is executed.

21.4.3.5 GPNVM Bit

GPNVM bits do not interfere with the embedded Flash memory plane. For more details, refer to Section 10. ”Memories” of this datasheet.

The ‘Set GPNVM Bit’ sequence is the following:

1. Execute the ‘Set GPNVM Bit’ command by writing EEFC_FCR.FCMD with the SGPB command and EEFC_FCR.FARG with the number of GPNVM bits to be set.
2. When the GPNVM bit is set, the bit EEFC_FSR.FRDY rises. If an interrupt was enabled by setting the bit EEFC_FMR.FRDY, the interrupt line of the interrupt controller is activated.
3. The result of the SGPB command can be checked by running a ‘Get GPNVM Bit’ (GGPB) command.

Note: The value of the FARG argument passed together with SGPB command must not exceed the higher GPNVM index available in the product. Flash data content is not altered if FARG exceeds the limit. Command Error is detected only if FARG is greater than 8.

Two errors can be detected in EEFC_FSR after a programming sequence:

- Command Error: A bad keyword has been written in EEFC_FCR.
- Flash Error: At the end of the programming, the EraseVerify or WriteVerify test of the Flash memory has failed.

It is possible to clear GPNVM bits previously set. The ‘Clear GPNVM Bit’ sequence is the following:

1. Execute the ‘Clear GPNVM Bit’ command by writing EEFC_FCR.FCMD with the CGPB command and EEFC_FCR.FARG with the number of GPNVM bits to be cleared.
2. When the clear completes, the bit EEFC_FSR.FRDY rises. If an interrupt has been enabled by setting the bit EEFC_FMR.FRDY, the interrupt line of the interrupt controller is activated.

Note: The value of the FARG argument passed together with CGPB command must not exceed the higher GPNVM index available in the product. Flash data content is not altered if FARG exceeds the limit. Command Error is detected only if FARG is greater than 8.

Two errors can be detected in EEFC_FSR after a programming sequence:

- Command Error: A bad keyword has been written in EEFC_FCR.
- Flash Error: At the end of the programming, the EraseVerify or WriteVerify test of the Flash memory has failed.

The status of GPNVM bits can be returned by the EEFC. The sequence is the following:

1. Execute the ‘Get GPNVM Bit’ command by writing EEFC_FCR.FCMD with the GGPB command. Field EEFC_FCR.FARG is meaningless.
2. GPNVM bits can be read by the software application in EEFC_FRR. The first word read corresponds to the 32 first GPNVM bits, following reads provide the next 32 GPNVM bits as long as it is meaningful. Extra reads to EEFC_FRR return 0.

Downloaded from Arrow.com.

For example, if the third bit of the first word read in EEFC_FRR is set, the third GPNVM bit is active.

One error can be detected in EEFC_FSR after a programming sequence:

- Command Error: A bad keyword has been written in EEFC_FCR.

Note: Access to the Flash in read is permitted when a ‘Set GPNVM Bit’, ‘Clear GPNVM Bit’ or ‘Get GPNVM Bit’ command is executed.

21.4.3.6 Calibration Bit

Calibration bits do not interfere with the embedded Flash memory plane.

The calibration bits cannot be modified.

The status of calibration bits are returned by the EEFC. The sequence is the following:

1. Execute the ‘Get CALIB Bit’ command by writing EEFC_FCR.FCMD with the GCALB command. Field EEFC_FCR.FARG is meaningless.
2. Calibration bits can be read by the software application in EEFC_FRR. The first word read corresponds to the first 32 calibration bits. The following reads provide the next 32 calibration bits as long as it is meaningful. Extra reads to EEFC_FRR return 0.

The 8/12 MHz fast RC oscillator is calibrated in production. This calibration can be read through the GCALB command. Table 21-5 shows the bit implementation.

The RC calibration for the 4 MHz is set to ‘1000000’.

Table 21-5. Calibration Bit Indexes

RC Calibration Frequency	EEFC_FRR Bits
8 MHz output	[28–22]
12 MHz output	[38–32]

21.4.3.7 Security Bit Protection

When the security bit is enabled, the Embedded Trace Macrocell (ETM) is disabled and access to the Flash through the SWD interface or through the Fast Flash Programming interface is forbidden. This ensures the confidentiality of the code programmed in the Flash.

The security bit is GPNVM0.

Disabling the security bit can only be achieved by asserting the ERASE pin at ‘1’, and after a full Flash erase is performed. When the security bit is deactivated, all accesses to the Flash are permitted.

21.4.3.8 Unique Identifier Area

Each device is programmed with a 128-bit unique identifier area. See Figure 21-1 "Flash Memory Areas".

The sequence to read the unique identifier area is the following:

1. Execute the ‘Start Read Unique Identifier’ command by writing EEFC_FCR.FCMD with the STUI command. Field EEFC_FCR.FARG is meaningless.
2. Wait until the bit EEFC_FSR.FRDY falls to read the unique identifier area. The unique identifier field is located in the first 128 bits of the Flash memory mapping. The ‘Start Read Unique Identifier’ command reuses some addresses of the memory plane for code, but the unique identifier area is physically different from the memory plane for code.
3. To stop reading the unique identifier area, execute the ‘Stop Read Unique Identifier’ command by writing EEFC_FCR.FCMD with the SPUI command. Field EEFC_FCR.FARG is meaningless.
4. When the SPUI command has been executed, the bit EEFC_FSR.FRDY rises. If an interrupt was enabled by setting the bit EEFC_FMR.FRDY, the interrupt line of the interrupt controller is activated.

Note that during the sequence, the software cannot be fetched from the Flash.

Downloaded from Arrow.com.

21.4.3.9 User Signature Area

Each product contains a user signature area of 512 bytes. It can be used for storage. Read, write and erase of this area is allowed.

See Figure 21-1 "Flash Memory Areas".

The sequence to read the user signature area is the following:

1. Execute the ‘Start Read User Signature’ command by writing EEFC_FCR.FCMD with the STUS command. Field EEFC_FCR.FARG is meaningless.
2. Wait until the bit EEFC_FSR.FRDY falls to read the user signature area. The user signature area is located in the first 512 bytes of the Flash memory mapping. The ‘Start Read User Signature’ command reuses some addresses of the memory plane but the user signature area is physically different from the memory plane
3. To stop reading the user signature area, execute the ‘Stop Read User Signature’ command by writing EEFC_FCR.FCMD with the SPUS command. Field EEFC_FCR.FARG is meaningless.
4. When the SPUI command has been executed, the bit EEFC_FSR.FRDY rises. If an interrupt was enabled by setting the bit EEFC_FMR.FRDY, the interrupt line of the interrupt controller is activated.

Note that during the sequence, the software cannot be fetched from the Flash or from the second plane in case of dual plane.

One error can be detected in EEFC_FSR after this sequence:

- Command Error: A bad keyword has been written in EEFC_FCR.

The sequence to write the user signature area is the following:

1. Write the full page, at any page address, within the internal memory area address space.
2. Execute the ‘Write User Signature’ command by writing EEFC_FCR.FCMD with the WUS command. Field EEFC_FCR.FARG is meaningless.
3. When programming is completed, the bit EEFC_FSR.FRDY rises. If an interrupt has been enabled by setting the bit EEFC_FMR.FRDY, the corresponding interrupt line of the interrupt controller is activated.

Two errors can be detected in EEFC_FSR after this sequence:

- Command Error: A bad keyword has been written in EEFC_FCR.
- Flash Error: At the end of the programming, the WriteVerify test of the Flash memory has failed.

The sequence to erase the user signature area is the following:

1. Execute the ‘Erase User Signature’ command by writing EEFC_FCR.FCMD with the EUS command. Field EEFC_FCR.FARG is meaningless.
2. When programming is completed, the bit EEFC_FSR.FRDY rises. If an interrupt has been enabled by setting the bit EEFC_FMR.FRDY, the corresponding interrupt line of the interrupt controller is activated.

Two errors can be detected in EEFC_FSR after this sequence:

- Command Error: A bad keyword has been written in EEFC_FCR.
- Flash Error: At the end of the programming, the EraseVerify test of the Flash memory has failed.

21.4.3.10 ECC Errors and Corrections

The Flash embeds an ECC module able to correct one unique error and able to detect two errors. The errors are detected while a read access is performed into memory array and stored in EEFC_FSR (see Section 21.5.3 ”EEFC Flash Status Register”). The error report is kept until EEFC_FSR is read.

There is one flag for a unique error on lower half part of the Flash word (64 LSB) and one flag for the upper half part (MSB). The multiple errors are reported in the same way.

Due to the anticipation technique to improve bandwidth throughput on instruction fetch, a reported error can be located in the next sequential Flash word compared to the location of the instruction being executed, which is located in the previously fetched Flash word.

Downloaded from Arrow.com.

If a software routine processes the error detection independently from the main software routine, the entire Flash located software must be rewritten because there is no storage of the error location.

If only a software routine is running to program and check pages by reading EEFC_FSR, the situation differs from the previous case. Performing a check for ECC unique errors just after page programming completion involves a read of the newly programmed page. This read sequence is viewed as data accesses and is not optimized by the Flash controller. Thus, in case of unique error, only the current page must be reprogrammed.

21.4.4 Register Write Protection

To prevent any single software error from corrupting EEFC behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the (EEFC_WPMR).

The following register can be write-protected:

- EEFC Flash Mode Register

Downloaded from Arrow.com.

21.5 Enhanced Embedded Flash Controller (EEFC) User Interface

The User Interface of the Embedded Flash Controller (EEFC) is integrated within the System Controller with base address 0x400E0C00.

Table 21-6. Register Mapping

Offset	Register	Name	Access	Reset State
0x00	EEFC Flash Mode Register	EEFC_FMR	Read/Write	0x0400_0000
0x04	EEFC Flash Command Register	EEFC_FCR	Write-only	–
0x08	EEFC Flash Status Register	EEFC_FSR	Read-only	0x0000_0001
0x0C	EEFC Flash Result Register	EEFC_FRR	Read-only	0x0
0x10–0x14	Reserved	–	–	–
0x18–0xE0	Reserved	–	–	–
0xE4	Write Protection Mode Register	EEFC_WPMR	Read/Write	0x0

Downloaded from Arrow.com.

21.5.1 EEFC Flash Mode Register

Name: EEFC_FMR

Address: 0x400E0C00

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	CLOE	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	SCOD
15	14	13	12	11	10	9	8
–	–	–	–	FWS			
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	FRDY

This register can only be written if the WPEN bit is cleared in the “EEFC Write Protection Mode Register” .

- **FRDY: Flash Ready Interrupt Enable**

0: Flash ready does not generate an interrupt.

1: Flash ready (to accept a new command) generates an interrupt.

- **FWS: Flash Wait State**

This field defines the number of wait states for read and write operations:

FWS = Number of cycles for Read/Write operations - 1

- **SCOD: Sequential Code Optimization Disable**

0: The sequential code optimization is enabled.

1: The sequential code optimization is disabled.

No Flash read should be done during change of this field.

- **CLOE: Code Loop Optimization Enable**

0: The opcode loop optimization is disabled.

1: The opcode loop optimization is enabled.

No Flash read should be done during change of this field.

Downloaded from Arrow.com.

21.5.2 EEFC Flash Command Register

Name: EEFC_FCR

Address: 0x400E0C04

Access: Write-only

31	30	29	28	27	26	25	24
FKEY							
23	22	21	20	19	18	17	16
FARG							
15	14	13	12	11	10	9	8
FARG							
7	6	5	4	3	2	1	0
FCMD							

- **FCMD: Flash Command**

Value	Name	Description
0x00	GETD	Get Flash descriptor
0x01	WP	Write page
0x02	WPL	Write page and lock
0x03	EWP	Erase page and write page
0x04	EWPL	Erase page and write page then lock
0x05	EA	Erase all
0x07	EPA	Erase pages
0x08	SLB	Set lock bit
0x09	CLB	Clear lock bit
0x0A	GLB	Get lock bit
0x0B	SGPB	Set GPNVM bit
0x0C	CGPB	Clear GPNVM bit
0x0D	GGPB	Get GPNVM bit
0x0E	STUI	Start read unique identifier
0x0F	SPUI	Stop read unique identifier
0x10	GCALB	Get CALIB bit
0x11	ES	Erase sector
0x12	WUS	Write user signature
0x13	EUS	Erase user signature
0x14	STUS	Start read user signature
0x15	SPUS	Stop read user signature

Downloaded from Arrow.com.

- **FARG: Flash Command Argument**

GETD, GLB, GGPB, STUI, SPUI, GCALB, WUS, EUS, STUS, SPUS, EA	Commands requiring no argument, including Erase all command	FARG is meaningless, must be written with 0
ES	Erase sector command	FARG must be written with any page number within the sector to be erased
EPA	Erase pages command	FARG[1:0] defines the number of pages to be erased The start page must be written in FARG[15:2]. FARG[1:0] = 0: Four pages to be erased. FARG[15:2] = Page_Number / 4 FARG[1:0] = 1: Eight pages to be erased. FARG[15:3] = Page_Number / 8, FARG[2]=0 FARG[1:0] = 2: Sixteen pages to be erased. FARG[15:4] = Page_Number / 16, FARG[3:2]=0 FARG[1:0] = 3: Thirty-two pages to be erased. FARG[15:5] = Page_Number / 32, FARG[4:2]=0 Refer to Table 21-4 “EEFC_FCR.FARG Field for EPA Command”.
WP, WPL, EWP, EWPL	Programming commands	FARG must be written with the page number to be programmed
SLB, CLB	Lock bit commands	FARG defines the page number to be locked or unlocked
SGPB, CGPB	GPNVM commands	FARG defines the GPNVM number to be programmed

- **FKEY: Flash Writing Protection Key**

Value	Name	Description
0x5A	PASSWD	The 0x5A value enables the command defined by the bits of the register. If the field is written with a different value, the write is not performed and no action is started.

Downloaded from Arrow.com.

21.5.3 EEFC Flash Status Register

Name: EEFC_FSR

Address: 0x400E0C08

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	MECCEMSB	UECCEMSB	MECCELSB	UECCELSB
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	FLERR	FLOCKE	FCMDE	FRDY

- **FRDY: Flash Ready Status (cleared when Flash is busy)**

0: The EEFC is busy.

1: The EEFC is ready to start a new command.

When set, this flag triggers an interrupt if the FRDY flag is set in EEFC_FMR.

This flag is automatically cleared when the EEFC is busy.

- **FCMDE: Flash Command Error Status (cleared on read or by writing EEFC_FCR)**

0: No invalid commands and no bad keywords were written in EEFC_FMR.

1: An invalid command and/or a bad keyword was/were written in EEFC_FMR.

- **FLOCKE: Flash Lock Error Status (cleared on read)**

0: No programming/erase of at least one locked region has happened since the last read of EEFC_FSR.

1: Programming/erase of at least one locked region has happened since the last read of EEFC_FSR.

This flag is automatically cleared when EEFC_FSR is read or EEFC_FCR is written.

- **FLERR: Flash Error Status (cleared when a programming operation starts)**

0: No Flash memory error occurred at the end of programming (EraseVerify or WriteVerify test has passed).

1: A Flash memory error occurred at the end of programming (EraseVerify or WriteVerify test has failed).

- **UECCELSB: Unique ECC Error on LSB Part of the Memory Flash Data Bus (cleared on read)**

0: No unique error detected on 64 LSB data bus of the Flash memory since the last read of EEFC_FSR.

1: One unique error detected but corrected on 64 LSB data bus of the Flash memory since the last read of EEFC_FSR.

- **MECCELSB: Multiple ECC Error on LSB Part of the Memory Flash Data Bus (cleared on read)**

0: No multiple error detected on 64 LSB part of the Flash memory data bus since the last read of EEFC_FSR.

1: Multiple errors detected and NOT corrected on 64 LSB part of the Flash memory data bus since the last read of EEFC_FSR.

- **UECCEMSB: Unique ECC Error on MSB Part of the Memory Flash Data Bus (cleared on read)**

0: No unique error detected on 64 MSB data bus of the Flash memory since the last read of EEFC_FSR.

1: One unique error detected but corrected on 64 MSB data bus of the Flash memory since the last read of EEFC_FSR.

Downloaded from Arrow.com.

- **MECCEMSB: Multiple ECC Error on MSB Part of the Memory Flash Data Bus (cleared on read)**

0: No multiple error detected on 64 MSB part of the Flash memory data bus since the last read of EEFC_FSR.

1: Multiple errors detected and NOT corrected on 64 MSB part of the Flash memory data bus since the last read of EEFC_FSR.

Downloaded from Arrow.com.

21.5.4 EEFC Flash Result Register

Name: EEFC_FRR

Address: 0x400E0C0C

Access: Read-only

31	30	29	28	27	26	25	24
FVALUE							
23	22	21	20	19	18	17	16
FVALUE							
15	14	13	12	11	10	9	8
FVALUE							
7	6	5	4	3	2	1	0
FVALUE							

- **FVALUE: Flash Result Value**

The result of a Flash command is returned in this register. If the size of the result is greater than 32 bits, the next resulting value is accessible at the next register read.

Downloaded from Arrow.com.

21.5.5 EEFC Write Protection Mode Register

Name: EEFC_WPMR

Address: 0x400E0CE4

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							

23	22	21	20	19	18	17	16
WPKEY							

15	14	13	12	11	10	9	8
WPKEY							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x454643 (EFC in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x454643 (EFC in ASCII).

See Section 21.4.4 ”Register Write Protection” for the list of registers that can be protected.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x454643	PASSWD	Writing any other value in this field aborts the write operation. Always reads as 0.

Downloaded from Arrow.com.

22. Supply Controller (SUPC)

22.1 Description

The Supply Controller (SUPC) controls the supply voltages of the system and manages the Backup mode. In this mode, current consumption is reduced to a few microamps for backup power retention. Exit from this mode is possible on multiple wake-up sources. The SUPC also generates the slow clock by selecting either the low-power RC oscillator or the low-power crystal oscillator.

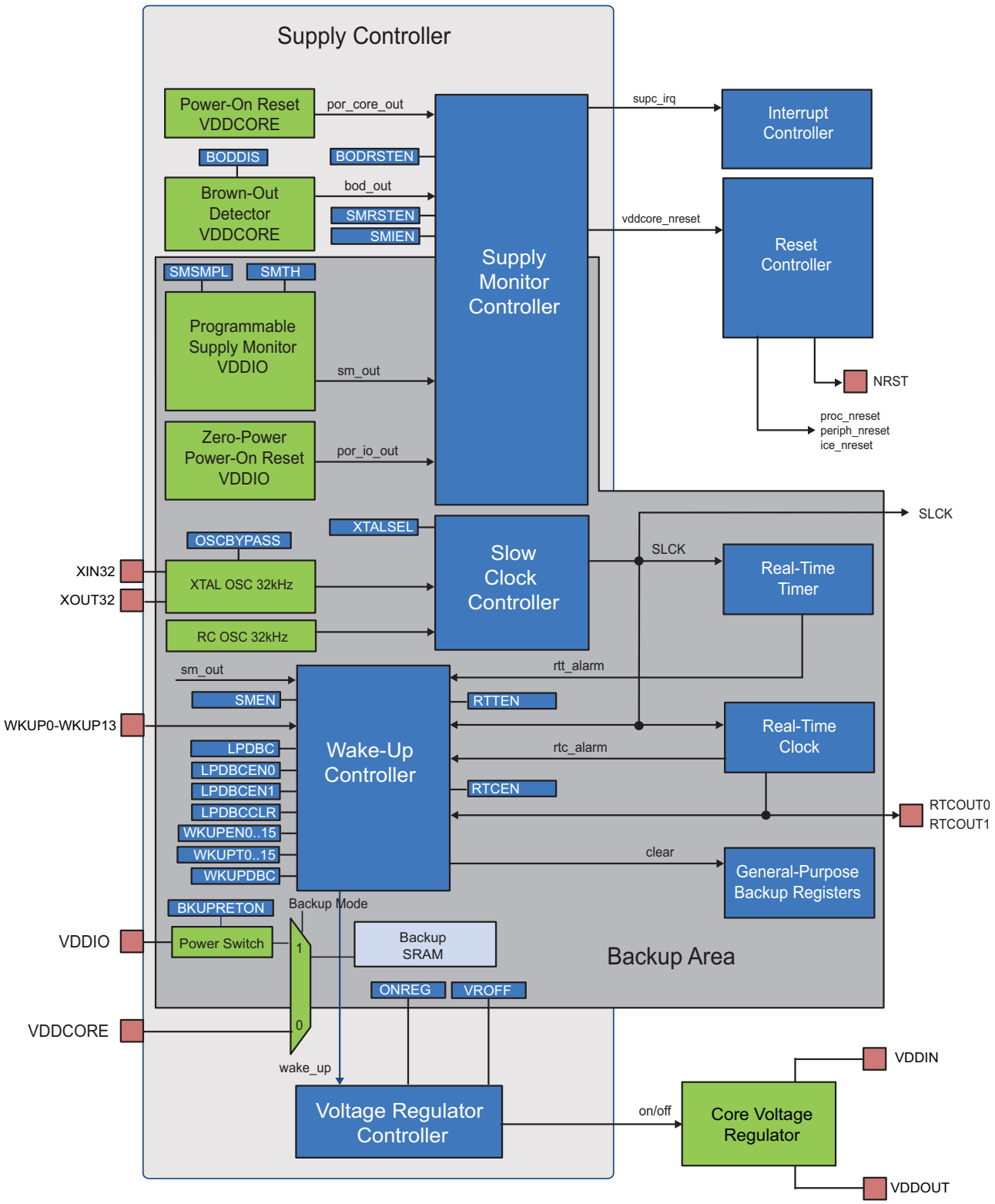
22.2 Embedded Characteristics

- Manages the core power supply VDDCORE and backup mode by controlling the embedded voltage regulator
- A supply monitor detection on VDDIO or a brownout detection on VDDCORE triggers a core reset
- Generates the slow clock SLCK by selecting either the 22-42 kHz low-power RC oscillator or the 32 kHz low-power crystal oscillator
- Backup SRAM
- Low-power tamper detection on two inputs
- Anti-tampering by immediate clear of the general-purpose backup registers
- Supports multiple wake-up sources for exit from backup mode
  - 14 Wake-up Inputs with programmable debouncing
  - Real-Time Clock Alarm
  - Real-Time Timer Alarm
  - Supply monitor detection on VDDIO, with programmable scan period and voltage threshold

Downloaded from Arrow.com.

22.3 Block Diagram

Figure 22-1. Supply Controller Block Diagram



Downloaded from Arrow.com.

22.4 Functional Description

22.4.1 Overview

The device is divided into two power supply areas:

- VDDIO power supply: includes the Supply Controller, part of the Reset Controller, the slow clock switch, the general-purpose backup registers, the supply monitor and the clock which includes the Real-time Timer and the Real-time Clock.
- Core power supply: includes part of the Reset Controller, the Brownout Detector, the processor, the SRAM memory, the Flash memory and the peripherals.

The Supply Controller (SUPC) controls the supply voltage of the core power supply. The SUPC intervenes when the VDDIO power supply rises (when the system is starting) or when Backup mode is entered.

The SUPC also integrates the slow clock generator, which is based on a 32 kHz crystal oscillator, and an embedded 32 kHz RC oscillator. The slow clock defaults to the RC oscillator, but the software can enable the crystal oscillator and select it as the slow clock source.

The SUPC and the VDDIO power supply have a reset circuitry based on a zero-power power-on reset cell. The zero-power power-on reset allows the SUPC to start correctly as soon as the VDDIO voltage becomes valid.

At start-up of the system, once the backup voltage VDDIO is valid and the embedded 32 kHz RC oscillator is stabilized, the SUPC starts up the core by sequentially enabling the internal voltage regulator. The SUPC waits until the core voltage VDDCORE is valid, then releases the reset signal of the core vddcore_nreset signal.

Once the system has started, the user can program a supply monitor and/or a brownout detector. If the supply monitor detects a voltage level on VDDIO that is too low, the SUPC asserts the reset signal of the core vddcore_nreset signal until VDDIO is valid. Likewise, if the brownout detector detects a core voltage level VDDCORE that is too low, the SUPC asserts the reset signal vddcore_nreset until VDDCORE is valid.

When Backup mode is entered, the SUPC sequentially asserts the reset signal of the core power supply vddcore_nreset and disables the voltage regulator, in order to supply only the VDDIO power supply. Current consumption is reduced to a few microamps for the backup part retention. Exit from this mode is possible on multiple wake-up sources including an event on WKUP pins, or a clock alarm. To exit this mode, the SUPC operates in the same way as system start-up.

Downloaded from Arrow.com.

22.4.2 Slow Clock Generator

The SUPC embeds a slow clock generator that is supplied with the VDDIO power supply. As soon as the VDDIO is supplied, both the crystal oscillator and the embedded RC oscillator are powered up, but only the embedded RC oscillator is enabled. When the RC oscillator is selected as the slow clock source, the slow clock stabilizes more quickly than when the crystal oscillator is selected.

The user can select the crystal oscillator to be the source of the slow clock, as it provides a more accurate frequency than the RC oscillator. The crystal oscillator is selected by setting the XTALSEL bit in the SUPC Control register (SUPC_CR). The following sequence must be used to switch from the RC oscillator to the crystal oscillator:

1. The PIO lines multiplexed with XIN32 and XOUT32 are configured to be driven by the oscillator.
2. The crystal oscillator is enabled.
3. A number of RC oscillator clock periods is counted to cover the start-up time of the crystal oscillator. Refer to Section 56. ”Electrical Characteristics” for information on 32 kHz crystal oscillator start-up time.
4. The slow clock is switched to the output of the crystal oscillator.
5. The RC oscillator is disabled to save power.

The switching time may vary depending on the RC oscillator clock frequency range. The switch of the slow clock source is glitch-free. The OSCSEL bit of the SUPC Status register (SUPC_SR) indicates when the switch sequence is finished.

Reverting to the RC oscillator as a slow clock source is only possible by shutting down the VDDIO power supply.

If the user does not need the crystal oscillator, the XIN32 and XOUT32 pins should be left unconnected.

The user can also set the crystal oscillator in Bypass mode instead of connecting a crystal. In this case, the user has to provide the external clock signal on XIN32. The input characteristics of the XIN32 pin are given in the section ‘Electrical Characteristics. To enter Bypass mode, the OSCBYPASS bit in the Mode register (SUPC_MR) must be set before setting XTALSEL.

22.4.3 Core Voltage Regulator Control/Backup Low-power Mode

The SUPC can be used to control the embedded voltage regulator.

The voltage regulator automatically adapts its quiescent current depending on the required load current. Refer to Section 56. ”Electrical Characteristics”.

The user can switch off the voltage regulator, and thus put the device in Backup mode, by writing a 1 to the VROFF bit in SUPC_CR.

Backup mode can also be entered by executing the WFE (Wait for Event) Cortex-M processor instruction with the SLEEPDEEP bit set to 1.

This asserts the vddcore_nreset signal after the write resynchronization time, which lasts two slow clock cycles (worst case). Once the vddcore_nreset signal is asserted, the processor and the peripherals are stopped one slow clock cycle before the core power supply shuts off.

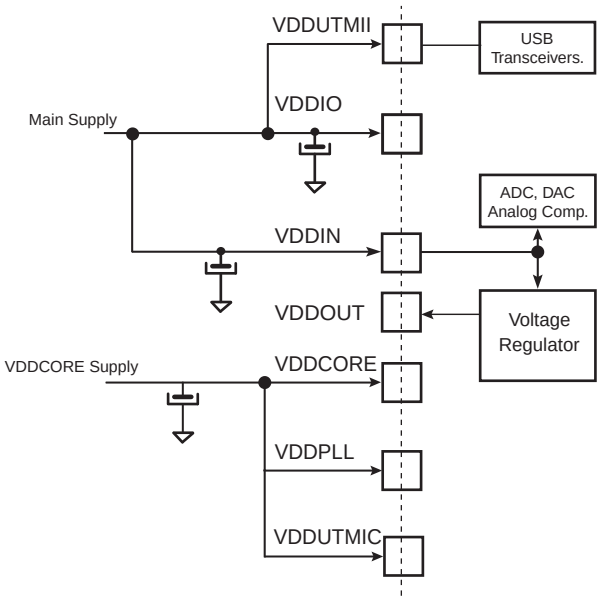
When the internal voltage regulator is not used and VDDCORE is supplied by an external supply, the voltage regulator can be disabled by writing a 1 to the ONREG bit in SUPC_MR.

Downloaded from Arrow.com.

22.4.4 Using Backup Batteries/Backup Supply

When backup batteries or, more generally, a separate backup supply is used, only VDDIO voltage is present in Backup mode. No other external supply is applied on the chip. In this case, the VDDIORDY bit in SUPC_MR must be cleared at least two slow clock periods before VDDIO voltage is removed. When waking up from Backup mode, VDDIORDY must be set.

Figure 22-2. Separate Backup Supply Powering Scheme

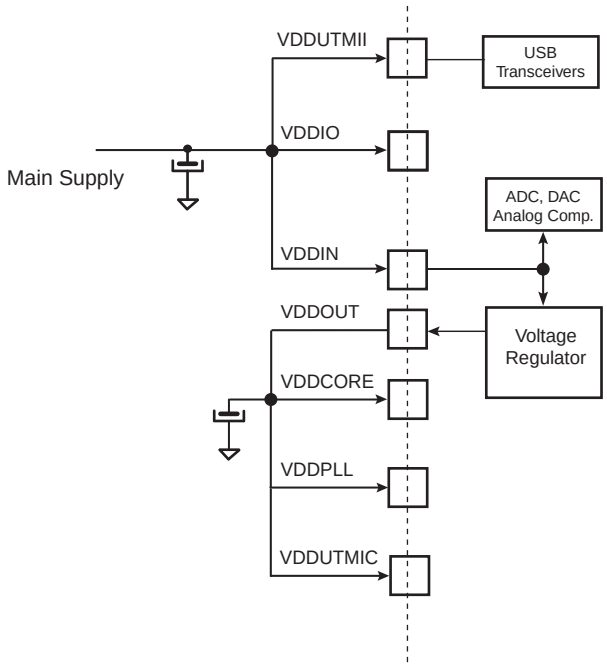


Note: Restrictions
With main supply < 3.0V, USB is not usable.
With main supply < 2.0V, ADC, DAC and Analog comparator are not usable.
With main supply and VDDIN > 3V, all peripherals are usable.

Downloaded from Arrow.com.

When no separate backup supply for VDDIO is used, since the external voltage applied on VDDIO is kept, all the I/O configurations (i.e., WKUP pin configuration) are maintained in Backup mode. When not using backup batteries, VDDIORDY is set so the user does not need to program it.

Figure 22-3. No Separate Backup Supply Powering Scheme

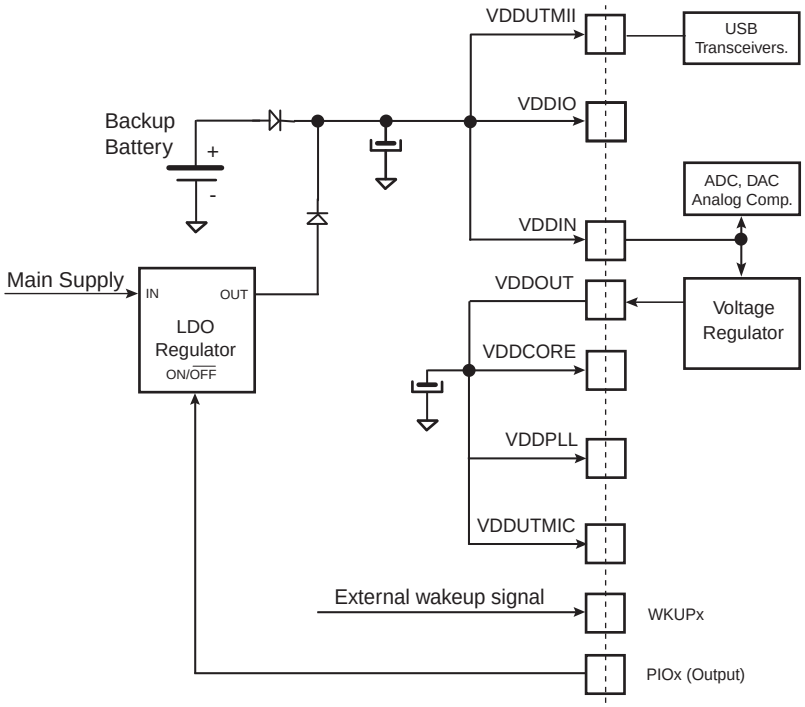


Note: Restrictions
With main supply < 2.0 V, USB and ADC/DAC and analog comparator are not usable.
With main supply > 2.0V and < 3V, USB is not usable.
With main supply > 3V, all peripherals are usable.

Downloaded from Arrow.com.

Figure 22-4 illustrates an example of the powering scheme when using a backup battery. Since the PIO state is preserved when in Backup mode, any free PIO line can be used to switch off the external regulator by driving the PIO line at low level (PIO is input, pull-up enabled after backup reset). System wake-up can be performed using a wake-up pin (WKUPx). See Section 22.4.9 "Wake-up Sources" for further details.

Figure 22-4. Battery Backup



Note: The two diodes provide a "switchover circuit" between the backup battery and the main supply when the system is put in backup mode.

22.4.5 Supply Monitor

The SUPC embeds a supply monitor located in the VDDIO power supply and which monitors VDDIO power supply.

The supply monitor can be used to prevent the processor from falling into an unpredictable state if the main power supply drops below a certain level.

The threshold of the supply monitor is programmable in the SMTH field of the Supply Monitor Mode register (SUPC_SMMR). Refer to Supply Monitor characteristics in Section 56. "Electrical Characteristics".

The supply monitor can also be enabled during one slow clock period on every one of either 32, 256 or 2048 slow clock periods, depending on the user selection. This is configured in the SMSMPL field in SUPC_SMMR.

Enabling the supply monitor for such reduced times divides the typical supply monitor power consumption by factors of 2, 16 and 128, respectively, if continuous monitoring of the VDDIO power supply is not required.

A supply monitor detection generates either a reset of the core power supply or a wake-up of the core power supply. Generating a core reset when a supply monitor detection occurs is enabled by setting the SMRSTEN bit in SUPC_SMMR.

Waking up the core power supply when a supply monitor detection occurs can be enabled by setting the SMEN bit in the Wake-up Mode register (SUPC_WUMR).

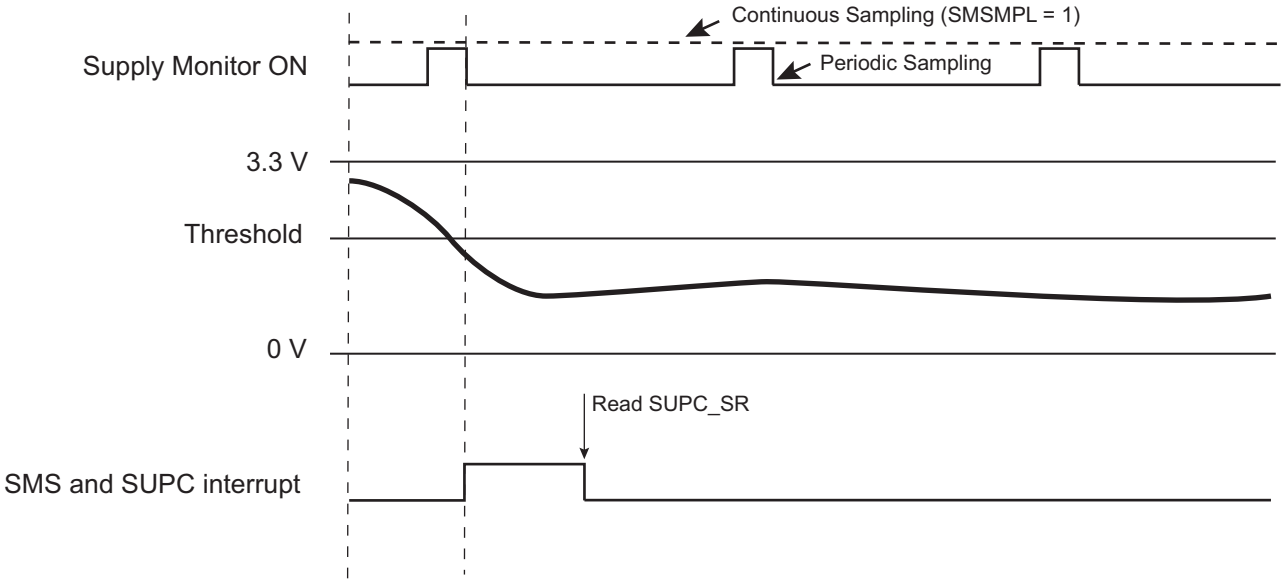
The SUPC provides two status bits in the SUPC_SR for the supply monitor that determine whether the last wake-up was due to the supply monitor:

Downloaded from Arrow.com.

- The SMOS bit provides real-time information, updated at each measurement cycle or updated at each slow clock cycle, if the measurement is continuous.
- The SMS bit provides saved information and shows a supply monitor detection has occurred since the last read of SUPC_SR.

The SMS flag generates an interrupt if the SMIEN bit is set in SUPC_SMMR.

Figure 22-5. Supply Monitor Status Bit and Associated Interrupt



22.4.6 Backup Power Supply Reset

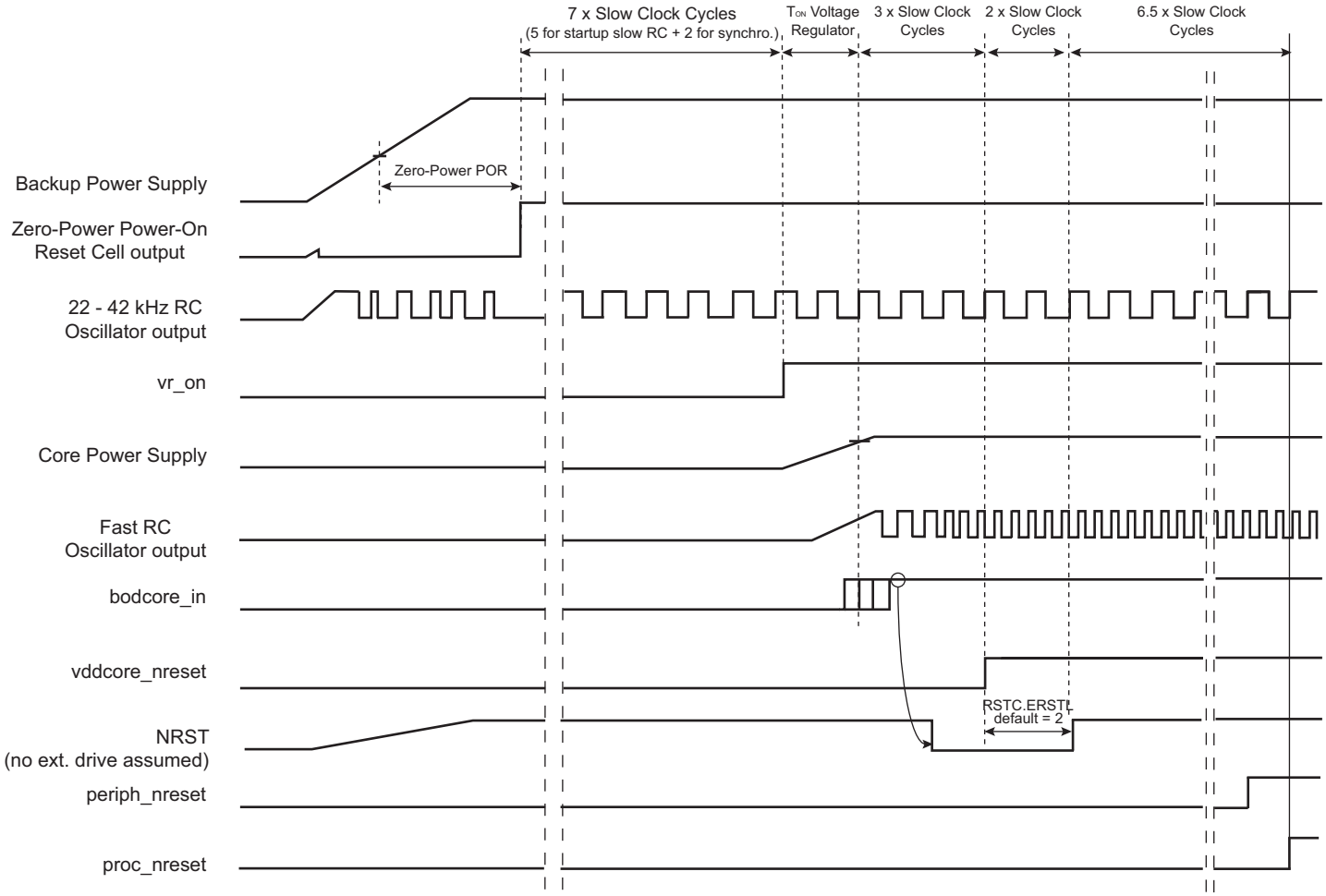
22.4.6.1 Raising the Backup Power Supply

When the backup voltage VDDIO rises, the RC oscillator is powered up and the zero-power power-on reset cell maintains its output low as long as VDDIO has not reached its target voltage. During this period, the SUPC is reset. When the VDDIO voltage becomes valid and the zero-power power-on reset signal is released, a counter is started for five slow clock cycles. This is the time required for the 32 kHz RC oscillator to stabilize.

After this time, the voltage regulator is enabled. The core power supply rises and the brownout detector provides the bodcore_in signal as soon as the core voltage VDDCORE is valid. This results in releasing the vddcore_nreset signal to the Reset Controller after the bodcore_in signal has been confirmed as being valid for at least one slow clock cycle.

Downloaded from Arrow.com.

Figure 22-6. Raising the VDDIO Power Supply



Note: After “proc_nreset” rising, the core starts fetching instructions from Flash at 4 MHz.

22.4.7 Core Reset

The Supply Controller manages the vddcore_nreset signal to the Reset Controller, as described in Section 22.4.6 ”Backup Power Supply Reset”. The vddcore_nreset signal is normally asserted before shutting down the core power supply and released as soon as the core power supply is correctly regulated.

There are two additional sources which can be programmed to activate vddcore_nreset:

- a supply monitor detection
- a brownout detection

22.4.7.1 Supply Monitor Reset

The supply monitor is capable of generating a reset of the system. This is enabled by setting the SMRSTEN bit in SUPC_SMMR.

If SMRSTEN is set and if a supply monitor detection occurs, the vddcore_nreset signal is immediately activated for a minimum of one slow clock cycle.

22.4.7.2 Brownout Detector Reset

The brownout detector provides the bodcore_in signal to the SUPC. This signal indicates that the voltage regulation is operating as programmed. If this signal is lost for longer than 1 slow clock period while the voltage regulator is enabled, the SUPC asserts vddcore_nreset if BODRSTEN is written to 1 in SUPC_MR.

Downloaded from Arrow.com.

If BODRSTEN is set and the voltage regulation is lost (output voltage of the regulator too low), the vddcore_nreset signal is asserted for a minimum of one slow clock cycle and then released if bodcore_in has been reactivated. The BODRSTS bit in SUPC_SR indicates the source of the last reset.

Until bodcore_in is deactivated, the vddcore_nreset signal remains active.

22.4.8 Controlling the SRAM Power Supply

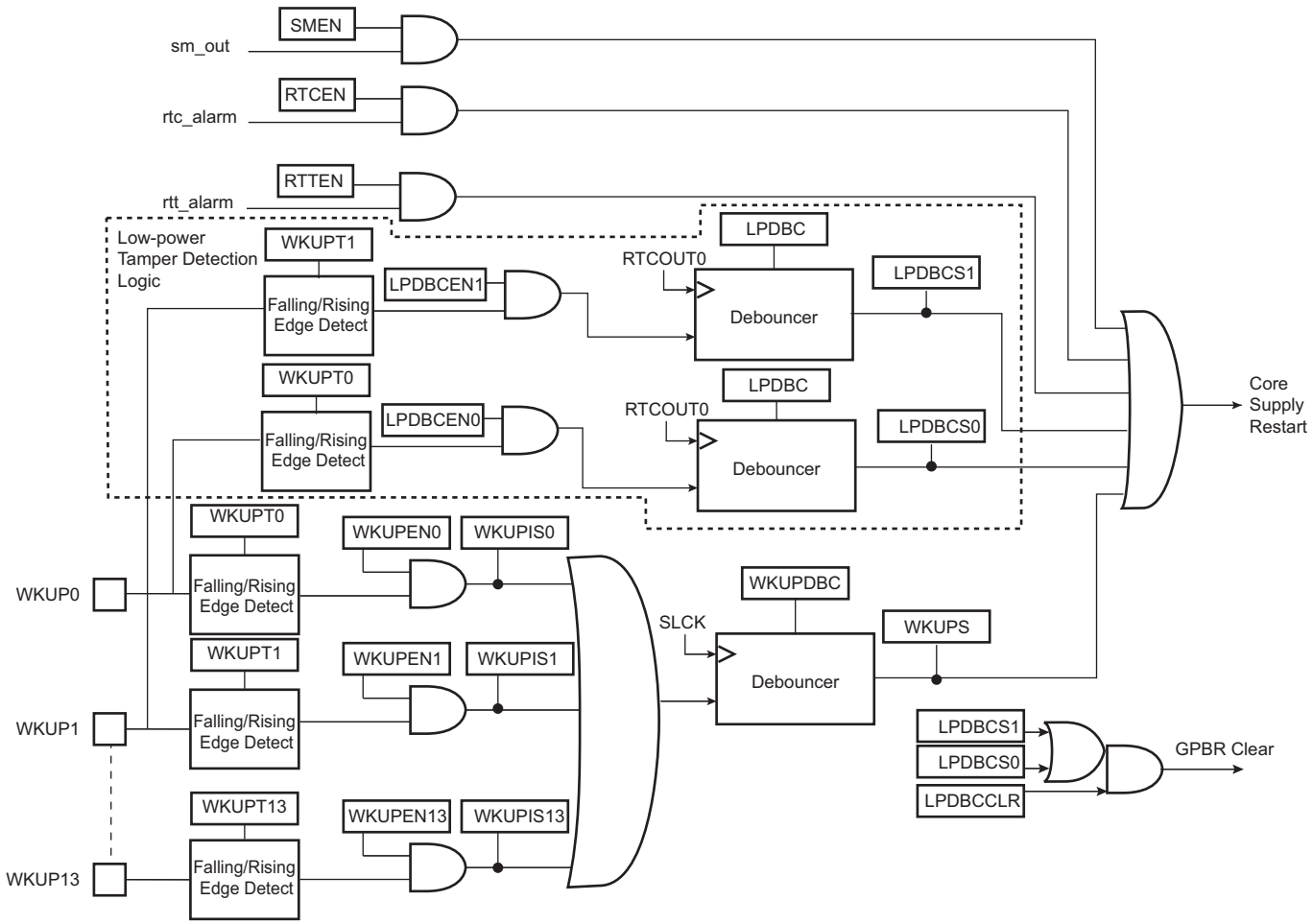
The SUPC can be used to switch on or off the power supply of the backup SRAM by opening or closing the SRAM power switch. This power switch is controlled by the BKUPRETON bit of SUPC_MR. However, the battery backup SRAM is automatically switched on when the core power supply is enabled, as the processor requires the SRAM as data memory space.

- If BKUPRETON is written to 1, there is no immediate effect, but the SRAM will be left powered when the SUPC enters Backup mode, thus retaining its content.
- If BKUPRETON is written to 0, there is no immediate effect, but the SRAM will be switched off when the SUPC enters Backup mode. The SRAM is automatically switched on when Backup mode is exited.

22.4.9 Wake-up Sources

The wake-up events allow the device to exit Backup mode. When a wake-up event is detected, the SUPC performs a sequence that automatically reenables the core power supply.

Figure 22-7. Wake-up Sources



Downloaded from Arrow.com.

22.4.9.1 Wake-up Inputs

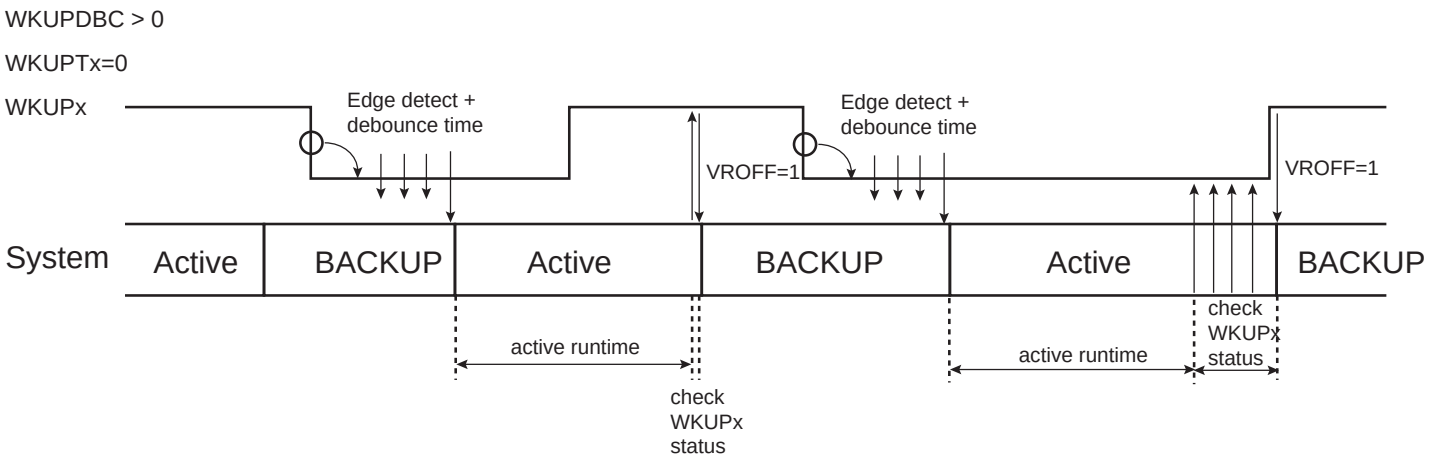
The wake-up inputs, WKUPx, can be programmed to perform a wake-up of the core power supply. Each input can be enabled by writing a 1 to the corresponding bit, WKUPENx, in the Wake-up Inputs register (SUPC_WUIR). The wake-up level can be selected with the corresponding polarity bit, WKUPTx, also located in SUPC_WUIR.

The resulting signals are wired-ORed to trigger a debounce counter, which is programmed with the WKUPDBC field in SUPC_WUMR. The WKUPDBC field selects a debouncing period of 3, 32, 512, 4,096 or 32,768 slow clock cycles. The duration of these periods corresponds, respectively, to about 100 µs, about 1 ms, about 16 ms, about 128 ms and about 1 second (for a typical slow clock frequency of 32 kHz). Programming WKUPDBC to 0x0 selects an immediate wake-up, i.e., an enabled WKUP pin must be active according to its polarity during a minimum of one slow clock period to wake up the core power supply.

If an enabled WKUP pin is asserted for a duration longer than the debouncing period, a wake-up of the core power supply is started and the signals, WKUP0 to WKUPx as shown in Figure 22-7 "Wake-up Sources", are latched in SUPC_SR. This allows the user to identify the source of the wake-up. However, if a new wake-up condition occurs, the primary information is lost. No new wake-up can be detected since the primary wake-up condition has disappeared.

Before instructing the system to enter Backup mode, if the field WKUPDBC > 0, it must be checked that none of the WKUPx pins that are enabled for a wake-up (exit from Backup mode) holds an active polarity. This is checked by reading the pin status in the PIO Controller. If WKUPENx=1 and the pin WKUPx holds an active polarity, the system must not be instructed to enter Backup mode.

Figure 22-8. Entering and Exiting Backup Mode with a WKUP Pin



22.4.9.2 Low-power Tamper Detection and Anti-Tampering

Low-power debouncer inputs (WKUP0, WKUP1) can be used for tamper detection. If the tamper sensor is biased through a resistor and constantly driven by the power supply, this leads to power consumption as long as the tamper detection switch is in its active state. To prevent power consumption when the switch is in active state, the tamper sensor circuitry must be intermittently powered, and thus a specific waveform must be applied to the sensor circuitry.

The waveform is generated using RTCOUTx in all modes including Backup mode. Refer to Section 26. "Real-time Clock (RTC)" for waveform generation.

Separate debouncers are embedded, one for WKUP0 input, one for WKUP1 input.

The WKUP0 and/or WKUP1 inputs perform a system wake-up upon tamper detection. This is enabled by setting the LPDBCEN0/1 bit in the SUPC_WUMR.

WKUP0 and/or WKUP1 inputs can also be used when VDDCORE is powered to detect a tamper.

Downloaded from Arrow.com.

When the bit LPDBCENx is written to 1, WKUPx pins must not be configured to act as a debouncing source for the WKUPDBC counter (WKUPENx must be cleared in SUPC_WUIR).

Low-power tamper detection or debounce requires RTC output (RTCOUTx) to be configured to generate a duty cycle programmable pulse (i.e., OUT0 = 0x7 in RTC_MR) in order to create the sampling points of both debouncers. The sampling point is the falling edge of the RTCOUTx waveform.

Figure 22-9 shows an example of an application where two tamper switches are used. RTCOUTx powers the external pull-up used by the tamper sensor circuitry.

Figure 22-9. Low-power Debouncer (Push-to-Make Switch, Pull-up Resistors)

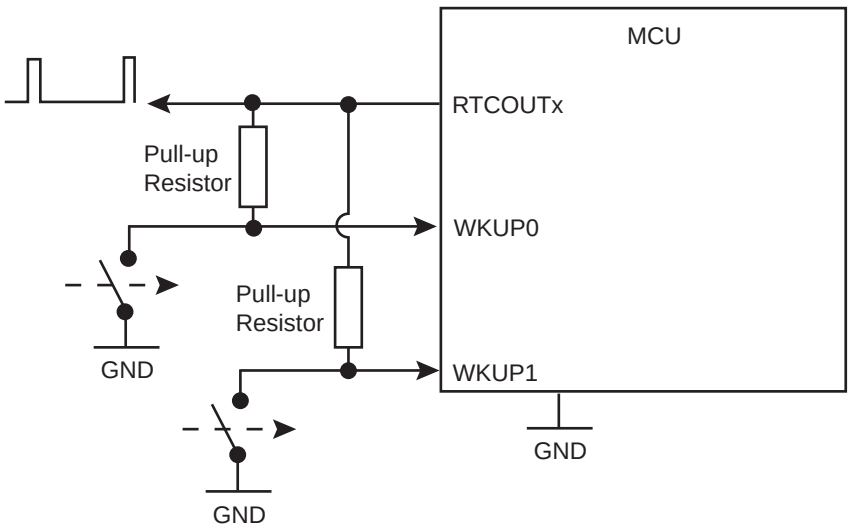
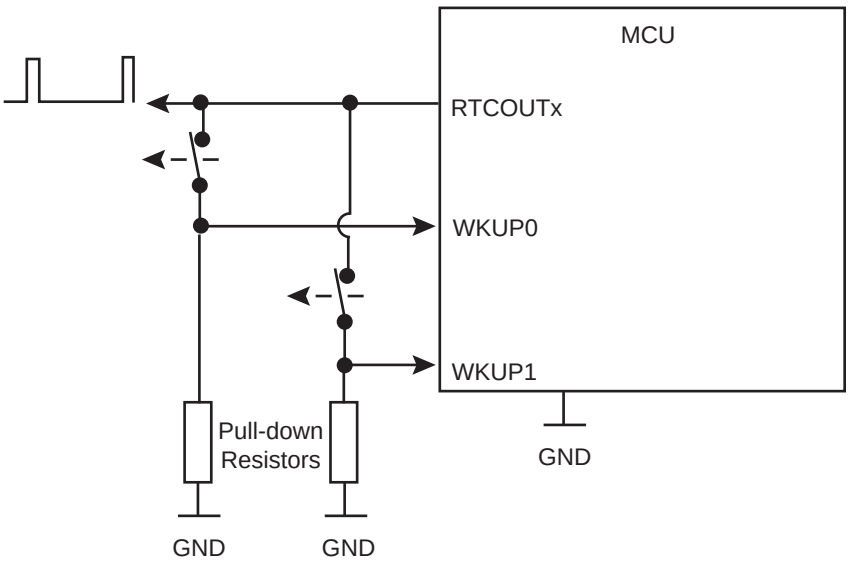


Figure 22-10. Low-power Debouncer (Push-to-Break Switch, Pull-down Resistors)



The debouncing period duration is configurable. The period is set for all debouncers (i.e., the duration cannot be adjusted for each debouncer). The number of successive identical samples to wake up the system can be configured from 2 up to 8 in the LPDBC field of SUPC_WUMR. The period of time between two samples can be configured by programming the TPERIOD field in the RTC_MR. Power parameters can be adjusted by modifying the period of time in the THIGH field in RTC_MR.

Downloaded from Arrow.com.

The wake-up polarity of the inputs can be independently configured by writing WKUPT0 and/ or WKUPT1 fields in SUPC_WUMR.

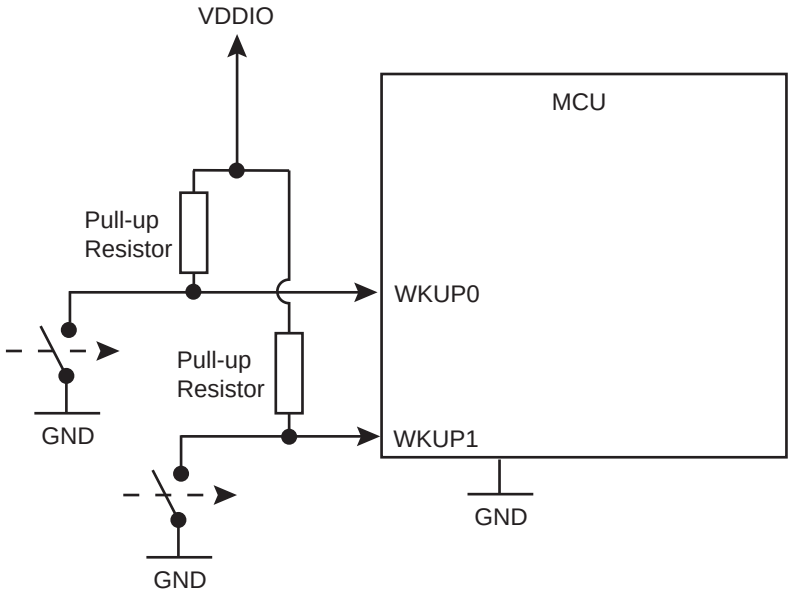
In order to determine which wake-up/tamper pin triggers the system wake-up, a status flag is associated for each low-power debouncer. These flags are read in SUPC_SR.

A debounce event (tamper detection) can perform an immediate clear (0 delay) on the first half the general-purpose backup registers (GPBR). The LPDBCCLR bit must be set in SUPC_WUMR.

Note that it is not mandatory to use the RTCOUTx pin when using the WKUP0/WKUP1 pins as tampering inputs in any mode. Using the RTCOUTx pin provides a “sampling mode” to further reduce the power consumption of the tamper detection circuitry. If RTCOUTx is not used, the RTC must be configured to create an internal sampling point for the debouncer logic. The period of time between two samples can be configured by programming the TPERIOD field in RTC_MR.

Figure 22-11 illustrates the use of WKUPx without the RTCOUTx pin.

Figure 22-11. Using WKUP Pins Without RTCOUTx Pins



22.4.9.3 Clock Alarms

The RTC and the RTT alarms can generate a wake-up of the core power supply. This can be enabled by setting, respectively, the bits RTCEN and RTTEN in SUPC_WUMR.

The Supply Controller does not provide any status as the information is available in the user interface of either the Real-Time Timer or the Real-Time Clock.

22.4.9.4 Supply Monitor Detection

The supply monitor can generate a wake-up of the core power supply. See Section 22.4.5 ”Supply Monitor”.

Downloaded from Arrow.com.

22.4.10 Register Write Protection

To prevent any single software error from corrupting SYSC behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the "System Controller Write Protection Mode Register" (SYSC_WPMR).

The following registers can be write-protected:

- RSTC Mode Register
- RTT Mode Register
- RTT Alarm Register
- RTC Control Register
- RTC Mode Register
- RTC Time Alarm Register
- RTC Calendar Alarm Register
- General Purpose Backup Registers
- Supply Controller Control Register
- Supply Controller Supply Monitor Mode Register
- Supply Controller Mode Register
- Supply Controller Wake-up Mode Register
- Supply Controller Wake-up Inputs Register

22.4.11 Register Bits in Backup Domain (VDDIO)

The following configuration registers, or certain bits of the registers, are physically located in the product backup domain:

- RSTC Mode Register (all bits)
- RTT Mode Register (all bits)
- RTT Alarm Register (all bits)
- RTC Control Register (all bits)
- RTC Mode Register (all bits)
- RTC Time Alarm Register (all bits)
- RTC Calendar Alarm Register (all bits)
- General Purpose Backup Registers (all bits)
- Supply Controller Control Register (see register description for details)
- Supply Controller Supply Monitor Mode Register (all bits)
- Supply Controller Mode Register (see register description for details)
- Supply Controller Wake-up Mode Register (all bits)
- Supply Controller Wake-up Inputs Register (all bits)
- Supply Controller Status Register (all bits)

Downloaded from Arrow.com.

22.5 Supply Controller (SUPC) User Interface

The user interface of the Supply Controller is part of the System Controller user interface.

22.5.1 System Controller (SYSC) User Interface

Table 22-1. System Controller Registers

Offset	System Controller Peripheral	Name
0x00-0x0c	Reset Controller	RSTC
0x10-0x2C	Supply Controller	SUPC
0x30-0x3C	Real Time Timer	RTT
0x50-0x5C	Watchdog Timer	WDT
0x60-0x8C	Real Time Clock	RTC
0x90-0xDC	General Purpose Backup Register	GPBR
0xE0	Reserved	–
0xE4	Write Protection Mode Register	SYSC_WPMR
0xE8-0xF8	Reserved	–

22.5.2 Supply Controller (SUPC) User Interface

Table 22-2. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Supply Controller Control Register	SUPC_CR	Write-only	–
0x04	Supply Controller Supply Monitor Mode Register	SUPC_SMMR	Read/Write	0x0000_0000
0x08	Supply Controller Mode Register	SUPC_MR	Read/Write	0x0000_5A00
0x0C	Supply Controller Wake-up Mode Register	SUPC_WUMR	Read/Write	0x0000_0000
0x10	Supply Controller Wake-up Inputs Register	SUPC_WUIR	Read/Write	0x0000_0000
0x14	Supply Controller Status Register	SUPC_SR	Read-only	0x0000_0000
0x18	Reserved	–	–	–

Downloaded from Arrow.com.

22.5.3 Supply Controller Control Register

Name: SUPC_CR

Address: 0x400E1810

Access: Write-only

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	XTALSEL	VROFF	–	–

This register can only be written if the WPEN bit is cleared in the System Controller Write Protection Mode Register (SYSC_MR).

- **VROFF: Voltage Regulator Off**

0 (NO_EFFECT): No effect.

1 (STOP_VREG): If KEY is correct, VROFF asserts the vddcore_nreset and stops the voltage regulator.

Note: This bit is located in the VDDIO domain.

- **XTALSEL: Crystal Oscillator Select**

0 (NO_EFFECT): No effect.

1 (CRYSTAL_SEL): If KEY is correct, XTALSEL switches the slow clock on the crystal oscillator output.

Note: This bit is located in the VDDIO domain.

- **KEY: Password**

Value	Name	Description
0xA5	PASSWD	Writing any other value in this field aborts the write operation.

Downloaded from Arrow.com.

22.5.4 Supply Controller Supply Monitor Mode Register

Name: SUPC_SMMR

Address: 0x400E1814

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	SMIEN	SMRSTEN	–	SMSMPL		
7	6	5	4	3	2	1	0
–	–	–	–	SMTH			

This register is located in the VDDIO domain.

This register can only be written if the WPEN bit is cleared in the System Controller Write Protection Mode Register (SYSC_MR).

- **SMTH: Supply Monitor Threshold**

Selects the threshold voltage of the supply monitor. Refer to Section 56. ”Electrical Characteristics” for voltage values.

- **SMSMPL: Supply Monitor Sampling Period**

Value	Name	Description
0x0	SMD	Supply Monitor disabled
0x1	CSM	Continuous Supply Monitor
0x2	32SLCK	Supply Monitor enabled one SLCK period every 32 SLCK periods
0x3	256SLCK	Supply Monitor enabled one SLCK period every 256 SLCK periods
0x4	2048SLCK	Supply Monitor enabled one SLCK period every 2,048 SLCK periods

- **SMRSTEN: Supply Monitor Reset Enable**

0 (NOT_ENABLE): The core reset signal vddcore_nreset is not affected when a supply monitor detection occurs.

1 (ENABLE): The core reset signal, vddcore_nreset is asserted when a supply monitor detection occurs.

- **SMIEN: Supply Monitor Interrupt Enable**

0 (NOT_ENABLE): The SUPC interrupt signal is not affected when a supply monitor detection occurs.

1 (ENABLE): The SUPC interrupt signal is asserted when a supply monitor detection occurs.

Downloaded from Arrow.com.

22.5.5 Supply Controller Mode Register

Name: SUPC_MR

Address: 0x400E1818

Access: Read/Write

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
–	–	–	OSCBYPASS	–	–	BKUPRETON	–

15	14	13	12	11	10	9	8
–	ONREG	BODDIS	BODRSTEN	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

This register can only be written if the WPEN bit is cleared in the System Controller Write Protection Mode Register (SYSC_MR).

- **BODRSTEN: Brownout Detector Reset Enable**

0 (NOT_ENABLE): The core reset signal vddcore_nreset is not affected when a brownout detection occurs.

1 (ENABLE): The core reset signal, vddcore_nreset is asserted when a brownout detection occurs.

Note: This bit is located in the VDDIO domain.

- **BODDIS: Brownout Detector Disable**

0 (ENABLE): The core brownout detector is enabled.

1 (DISABLE): The core brownout detector is disabled.

Note: This bit is located in the VDDIO domain.

- **ONREG: Voltage Regulator Enable**

0 (ONREG_UNUSED): Internal voltage regulator is not used (external power supply is used).

1 (ONREG_USED): Internal voltage regulator is used.

Note: This bit is located in the VDDIO domain.

- **BKUPRETON: SRAM On In Backup Mode**

0: SRAM (Backup) switched off in Backup mode.

1: SRAM (Backup) switched on in Backup mode.

Note: This bit is located in the VDDIO domain.

- **OSCBYPASS: Oscillator Bypass**

0 (NO_EFFECT): No effect. Clock selection depends on the value of XTALSEL (SUPC_CR).

1 (BYPASS): The 32 kHz crystal oscillator is bypassed if XTALSEL (SUPC_CR) is set. OSCBYPASS must be set prior to setting XTALSEL.

Note: This bit is located in the VDDIO domain.

Downloaded from Arrow.com.

- **KEY: Password Key**

Value	Name	Description
0xA5	PASSWD	Writing any other value in this field aborts the write operation.

Downloaded from Arrow.com.

22.5.6 Supply Controller Wake-up Mode Register

Name: SUPC_WUMR

Address: 0x400E181C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	LPDBC		

15	14	13	12	11	10	9	8
–	WKUPDBC			–	–	–	–

7	6	5	4	3	2	1	0
LPDBCCLR	LPDBCEN1	LPDBCEN0	–	RTCEN	RTTEN	SMEN	–

This register is located in the VDDIO domain.

This register can only be written if the WPEN bit is cleared in the System Controller Write Protection Mode Register (SYSC_MR).

- **SMEN: Supply Monitor Wake-up Enable**

0 (NOT_ENABLE): The supply monitor detection has no wake-up effect.

1 (ENABLE): The supply monitor detection forces the wake-up of the core power supply.

- **RTTEN: Real-time Timer Wake-up Enable**

0 (NOT_ENABLE): The RTT alarm signal has no wake-up effect.

1 (ENABLE): The RTT alarm signal forces the wake-up of the core power supply.

- **RTCEN: Real-time Clock Wake-up Enable**

0 (NOT_ENABLE): The RTC alarm signal has no wake-up effect.

1 (ENABLE): The RTC alarm signal forces the wake-up of the core power supply.

- **LPDBCEN0: Low-power Debouncer Enable WKUP0**

0 (NOT_ENABLE): The WKUP0 input pin is not connected to the low-power debouncer.

1 (ENABLE): The WKUP0 input pin is connected to the low-power debouncer and forces a system wake-up.

- **LPDBCEN1: Low-power Debouncer Enable WKUP1**

0 (NOT_ENABLE): The WKUP1 input pin is not connected to the low-power debouncer.

1 (ENABLE): The WKUP1 input pin is connected to the low-power debouncer and forces a system wake-up.

- **LPDBCCLR: Low-power Debouncer Clear**

0 (NOT_ENABLE): A low-power debounce event does not create an immediate clear on the first half of GPBR registers.

1 (ENABLE): A low-power debounce event on WKUP0 or WKUP1 generates an immediate clear on the first half of GPBR registers.

Downloaded from Arrow.com.

- **WKUPDBC: Wake-up Inputs Debouncer Period**

Value	Name	Description
0	IMMEDIATE	Immediate, no debouncing, detected active at least on one Slow Clock edge.
1	3_SLCK	WKUPx shall be in its active state for at least 3 SLCK periods
2	32_SLCK	WKUPx shall be in its active state for at least 32 SLCK periods
3	512_SLCK	WKUPx shall be in its active state for at least 512 SLCK periods
4	4096_SLCK	WKUPx shall be in its active state for at least 4,096 SLCK periods
5	32768_SLCK	WKUPx shall be in its active state for at least 32,768 SLCK periods

- **LPDBC: Low-power Debouncer Period**

Value	Name	Description
0	DISABLE	Disable the low-power debouncers.
1	2_RTCOUT	WKUP0/1 in active state for at least 2 RTCOUTx clock periods
2	3_RTCOUT	WKUP0/1 in active state for at least 3 RTCOUTx clock periods
3	4_RTCOUT	WKUP0/1 in active state for at least 4 RTCOUTx clock periods
4	5_RTCOUT	WKUP0/1 in active state for at least 5 RTCOUTx clock periods
5	6_RTCOUT	WKUP0/1 in active state for at least 6 RTCOUTx clock periods
6	7_RTCOUT	WKUP0/1 in active state for at least 7 RTCOUTx clock periods
7	8_RTCOUT	WKUP0/1 in active state for at least 8 RTCOUTx clock periods

Downloaded from Arrow.com.

22.5.7 Supply Controller Wake-up Inputs Register

Name: SUPC_WUIR

Address: 0x400E1820

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	WKUPT13	WKUPT12	WKUPT11	WKUPT10	WKUPT9	WKUPT8
23	22	21	20	19	18	17	16
WKUPT7	WKUPT6	WKUPT5	WKUPT4	WKUPT3	WKUPT2	WKUPT1	WKUPT0
15	14	13	12	11	10	9	8
–	–	WKUPEN13	WKUPEN12	WKUPEN11	WKUPEN10	WKUPEN9	WKUPEN8
7	6	5	4	3	2	1	0
WKUPEN7	WKUPEN6	WKUPEN5	WKUPEN4	WKUPEN3	WKUPEN2	WKUPEN1	WKUPEN0

This register is located in the VDDIO domain.

This register can only be written if the WPEN bit is cleared in the System Controller Write Protection Mode Register (SYSC_MR).

- **WKUPEN0 - WKUPENx: Wake-up Input Enable 0 to x**

0 (DISABLE): The corresponding wake-up input has no wake-up effect.

1 (ENABLE): The corresponding wake-up input is enabled for a wake-up of the core power supply.

- **WKUPT0 - WKUPTx: Wake-up Input Type 0 to x**

0 (LOW): A falling edge followed by a low level for a period defined by WKUPDBC on the corresponding wake-up input forces the wake-up of the core power supply.

1 (HIGH): A rising edge followed by a high level for a period defined by WKUPDBC on the corresponding wake-up input forces the wake-up of the core power supply.

Downloaded from Arrow.com.

22.5.8 Supply Controller Status Register

Name: SUPC_SR

Address: 0x400E1824

Access: Read-only

31	30	29	28	27	26	25	24
–	–	WKUPIS13	WKUPIS12	WKUPIS11	WKUPIS10	WKUPIS9	WKUPIS8
23	22	21	20	19	18	17	16
WKUPIS7	WKUPIS6	WKUPIS5	WKUPIS4	WKUPIS3	WKUPIS2	WKUPIS1	WKUPIS0
15	14	13	12	11	10	9	8
–	LPDBCS1	LPDBCS0	–	–	–	–	–
7	6	5	4	3	2	1	0
OSCSEL	SMOS	SMS	SMRSTS	BODRSTS	SMWS	WKUPS	–

Note: Because of the asynchronism between the Slow Clock (SLCK) and the System Clock (MCK), the status register flag reset is taken into account only 2 slow clock cycles after the read of the SUPC_SR.

This register is located in the VDDIO domain.

- **WKUPS: WKUP Wake-up Status (cleared on read)**

0 (NO): No wake-up due to the assertion of the WKUP pins has occurred since the last read of SUPC_SR.

1 (PRESENT): At least one wake-up due to the assertion of the WKUP pins has occurred since the last read of SUPC_SR.

- **SMWS: Supply Monitor Detection Wake-up Status (cleared on read)**

0 (NO): No wake-up due to a supply monitor detection has occurred since the last read of SUPC_SR.

1 (PRESENT): At least one wake-up due to a supply monitor detection has occurred since the last read of SUPC_SR.

- **BODRSTS: Brownout Detector Reset Status (cleared on read)**

0 (NO): No core brownout rising edge event has been detected since the last read of the SUPC_SR.

1 (PRESENT): At least one brownout output rising edge event has been detected since the last read of the SUPC_SR.

When the voltage remains below the defined threshold, there is no rising edge event at the output of the brownout detection cell. The rising edge event occurs only when there is a voltage transition below the threshold.

- **SMRSTS: Supply Monitor Reset Status (cleared on read)**

0 (NO): No supply monitor detection has generated a core reset since the last read of the SUPC_SR.

1 (PRESENT): At least one supply monitor detection has generated a core reset since the last read of the SUPC_SR.

- **SMS: Supply Monitor Status (cleared on read)**

0 (NO): No supply monitor detection since the last read of SUPC_SR.

1 (PRESENT): At least one supply monitor detection since the last read of SUPC_SR.

- **SMOS: Supply Monitor Output Status**

0 (HIGH): The supply monitor detected VDDIO higher than its threshold at its last measurement.

1 (LOW): The supply monitor detected VDDIO lower than its threshold at its last measurement.

Downloaded from Arrow.com.

- **OSCSEL: 32-kHz Oscillator Selection Status**

0 (RC): The slow clock, SLCK, is generated by the embedded 32 kHz RC oscillator.

1 (CRYST): The slow clock, SLCK, is generated by the 32 kHz crystal oscillator.

- **LPDBCS0: Low-power Debouncer Wake-up Status on WKUP0 (cleared on read)**

0 (NO): No wake-up due to the assertion of the WKUP0 pin has occurred since the last read of SUPC_SR.

1 (PRESENT): At least one wake-up due to the assertion of the WKUP0 pin has occurred since the last read of SUPC_SR.

- **LPDBCS1: Low-power Debouncer Wake-up Status on WKUP1 (cleared on read)**

0 (NO): No wake-up due to the assertion of the WKUP1 pin has occurred since the last read of SUPC_SR.

1 (PRESENT): At least one wake-up due to the assertion of the WKUP1 pin has occurred since the last read of SUPC_SR.

- **WKUPISx: WKUPx Input Status (cleared on read)**

0 (DIS): The corresponding wake-up input is disabled, or was inactive at the time the debouncer triggered a wake-up event.

1 (EN): The corresponding wake-up input was active at the time the debouncer triggered a wake-up event since the last read of SUPC_SR.

Downloaded from Arrow.com.

22.5.9 System Controller Write Protection Mode Register

Name: SYSC_WPMR

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							

23	22	21	20	19	18	17	16
WPKEY							

15	14	13	12	11	10	9	8
WPKEY							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x525443 (“RTC” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x525443 (“RTC” in ASCII).

See Section 22.4.10 ”Register Write Protection” for the list of registers that can be write-protected.

- **WPKEY: Write Protection Key**.

Value	Name	Description
0x525443	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

Downloaded from Arrow.com.

23. Watchdog Timer (WDT)

23.1 Description

The Watchdog Timer (WDT) is used to prevent system lock-up if the software becomes trapped in a deadlock. It features a 12-bit down counter that allows a watchdog period of up to 16 seconds (slow clock around 32 kHz). It can generate a general reset or a processor reset only. In addition, it can be stopped while the processor is in Debug mode or Sleep mode (Idle mode).

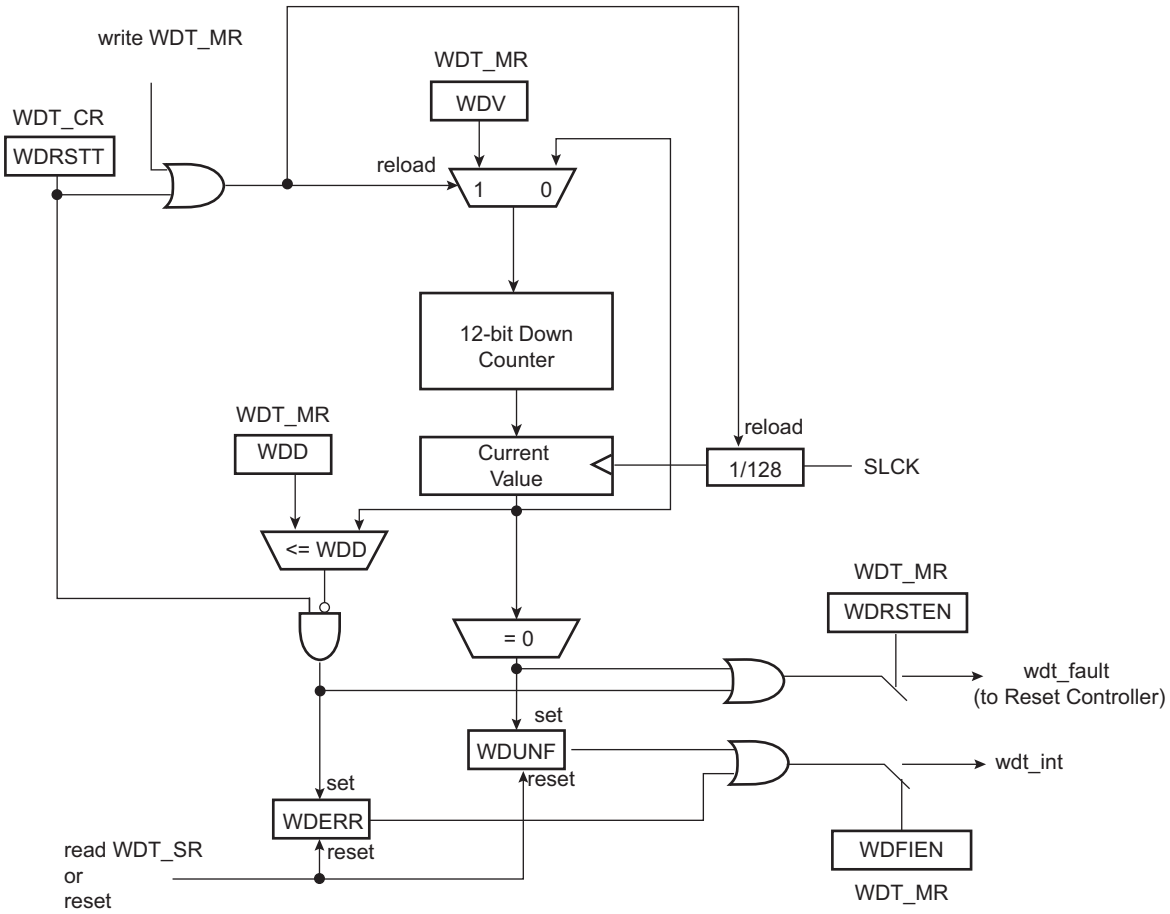
23.2 Embedded Characteristics

- 12-bit Key-protected Programmable Counter
- Watchdog Clock is Independent from Processor Clock
- Provides Reset or Interrupt Signals to the System
- Counter May Be Stopped while the Processor is in Debug State or in Idle Mode

Downloaded from Arrow.com.

23.3 Block Diagram

Figure 23-1. Watchdog Timer Block Diagram



Downloaded from Arrow.com.

23.4 Functional Description

The Watchdog Timer is used to prevent system lock-up if the software becomes trapped in a deadlock. It is supplied with VDDCORE. It restarts with initial values on processor reset.

The watchdog is built around a 12-bit down counter, which is loaded with the value defined in the field WDV of the Mode Register (WDT_MR). The Watchdog Timer uses the slow clock divided by 128 to establish the maximum watchdog period to be 16 seconds (with a typical slow clock of 32.768 kHz).

After a processor reset, the value of WDV is 0xFFF, corresponding to the maximum value of the counter with the external reset generation enabled (field WDRSTEN at 1 after a backup reset). This means that a default watchdog is running at reset, i.e., at power-up. The user can either disable the WDT by setting bit WDT_MR.WDDIS or reprogram the WDT to meet the maximum watchdog period the application requires.

When setting the WDDIS bit, and while it is set, the fields WDV and WDD must not be modified.

If the watchdog is restarted by writing into the Control Register (WDT_CR), WDT_MR must not be programmed during a period of time of three slow clock periods following the WDT_CR write access. In any case, programming a new value in WDT_MR automatically initiates a restart instruction.

WDT_MR can be written only once. Only a processor reset resets it. Writing WDT_MR reloads the timer with the newly programmed mode parameters.

In normal operation, the user reloads the watchdog at regular intervals before the timer underflow occurs, by setting bit WDT_CR.WDRSTT. The watchdog counter is then immediately reloaded from WDT_MR and restarted, and the slow clock 128 divider is reset and restarted. WDT_CR is write-protected. As a result, writing WDT_CR without the correct hard-coded key has no effect. If an underflow does occur, the “wdt_fault” signal to the Reset Controller is asserted if bit WDT_MR.WDRSTEN is set. Moreover, the bit WDUNF is set in the Status Register (WDT_SR).

To prevent a software deadlock that continuously triggers the watchdog, the reload of the watchdog must occur while the watchdog counter is within a window between 0 and WDD. WDD is defined in WDT_MR.

Any attempt to restart the watchdog while the watchdog counter is between WDV and WDD results in a watchdog error, even if the watchdog is disabled. The bit WDT_SR.WDERR is updated and the “wdt_fault” signal to the Reset Controller is asserted.

Note that this feature can be disabled by programming a WDD value greater than or equal to the WDV value. In such a configuration, restarting the Watchdog Timer is permitted in the whole range [0; WDV] and does not generate an error. This is the default configuration on reset (the WDD and WDV values are equal).

The status bits WDUNF (Watchdog Underflow) and WDERR (Watchdog Error) trigger an interrupt, provided the bit WDT_MR.WDFIEN is set. The signal “wdt_fault” to the Reset Controller causes a watchdog reset if the WDRSTEN bit is set as already explained in the Reset Controller documentation. In this case, the processor and the Watchdog Timer are reset, and the WDERR and WDUNF flags are reset.

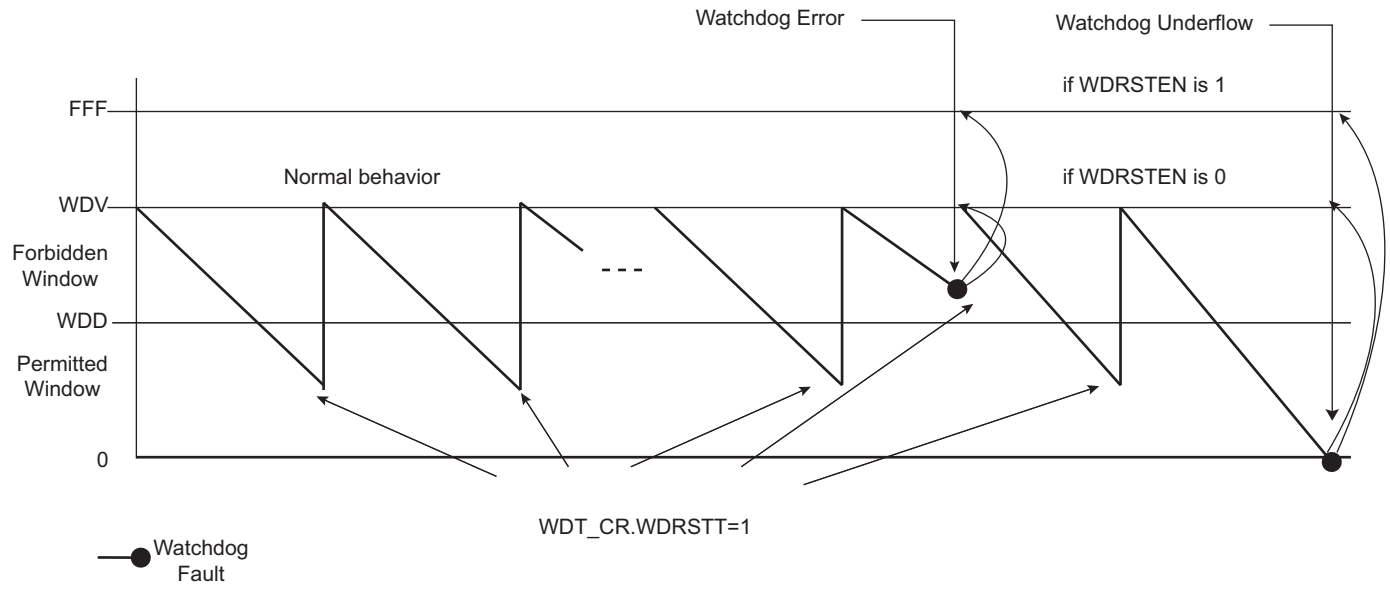
If a reset is generated or if WDT_SR is read, the status bits are reset, the interrupt is cleared, and the “wdt_fault” signal to the reset controller is deasserted.

Writing WDT_MR reloads and restarts the down counter.

While the processor is in debug state or in Sleep mode, the counter may be stopped depending on the value programmed for the bits WDIDLEHLT and WDDBGHLT in WDT_MR.

Downloaded from Arrow.com.

Figure 23-2. Watchdog Behavior



Downloaded from Arrow.com.

23.5 Watchdog Timer (WDT) User Interface

Table 23-1. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	WDT_CR	Write-only	–
0x04	Mode Register	WDT_MR	Read/Write Once	0x3FFF_2FFF
0x08	Status Register	WDT_SR	Read-only	0x0000_0000

Downloaded from Arrow.com.

23.5.1 Watchdog Timer Control Register

Name: WDT_CR

Address: 0x400E1850

Access: Write-only

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WDRSTT

Note: The WDT_CR register values must not be modified within three slow clock periods following a restart of the watchdog performed by a write access in WDT_CR. Any modification will cause the watchdog to trigger an end of period earlier than expected.

- **WDRSTT: Watchdog Restart**

0: No effect.

1: Restarts the watchdog if KEY is written to 0xA5.

- **KEY: Password**

Value	Name	Description
0xA5	PASSWD	Writing any other value in this field aborts the write operation.

Downloaded from Arrow.com.

23.5.2 Watchdog Timer Mode Register

Name: WDT_MR

Address: 0x400E1854

Access: Read/Write Once

31	30	29	28	27	26	25	24
–	–	WDIDLEHLT	WDDBGHLT	WDD			
23	22	21	20	19	18	17	16
WDD							
15	14	13	12	11	10	9	8
WDDIS	–	WDRSTEN	WDFIEN	WDV			
7	6	5	4	3	2	1	0
WDV							

Notes:
1. The first write access prevents any further modification of the value of this register. Read accesses remain possible.
2. The WDT_MR register values must not be modified within three slow clock periods following a restart of the watchdog performed by a write access in WDT_CR. Any modification will cause the watchdog to trigger an end of period earlier than expected.

- **WDV: Watchdog Counter Value**

Defines the value loaded in the 12-bit watchdog counter.

- **WDFIEN: Watchdog Fault Interrupt Enable**

0: A watchdog fault (underflow or error) has no effect on interrupt.

1: A watchdog fault (underflow or error) asserts interrupt.

- **WDRSTEN: Watchdog Reset Enable**

0: A watchdog fault (underflow or error) has no effect on the resets.

1: A watchdog fault (underflow or error) triggers a watchdog reset.

- **WDDIS: Watchdog Disable**

0: Enables the Watchdog Timer.

1: Disables the Watchdog Timer.

Note: When setting the WDDIS bit, and while it is set, the fields WDV and WDD must not be modified.

- **WDD: Watchdog Delta Value**

Defines the permitted range for reloading the Watchdog Timer.

If the Watchdog Timer value is less than or equal to WDD, setting bit WDT_CR.WDRSTT restarts the timer.

If the Watchdog Timer value is greater than WDD, setting bit WDT_CR.WDRSTT causes a watchdog error.

- **WDDBGHLT: Watchdog Debug Halt**

0: The watchdog runs when the processor is in debug state.

1: The watchdog stops when the processor is in debug state.

Downloaded from Arrow.com.

- **WDIDLEHLT: Watchdog Idle Halt**

0: The watchdog runs when the system is in idle state.

1: The watchdog stops when the system is in idle state.

Downloaded from Arrow.com.

23.5.3 Watchdog Timer Status Register

Name: WDT_SR

Address: 0x400E1858

Access Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	WDERR	WDUNF

- **WDUNF: Watchdog Underflow (cleared on read)**

0: No watchdog underflow occurred since the last read of WDT_SR.

1: At least one watchdog underflow occurred since the last read of WDT_SR.

- **WDERR: Watchdog Error (cleared on read)**

0: No watchdog error occurred since the last read of WDT_SR.

1: At least one watchdog error occurred since the last read of WDT_SR.

Downloaded from Arrow.com.

24. Reinforced Safety Watchdog Timer (RSWDT)

24.1 Description

The Reinforced Safety Watchdog Timer (RSWDT) works in parallel with the Watchdog Timer (WDT) to reinforce safe watchdog operations.

The RSWDT can be used to reinforce the safety level provided by the WDT in order to prevent system lock-up if the software becomes trapped in a deadlock. The RSWDT works in a fully operable mode, independent of the WDT. Its clock source is automatically selected from either the slow RC oscillator clock or main RC oscillator divided clock to get an equivalent slow RC oscillator clock. If the WDT clock source (for example, the 32 kHz crystal oscillator) fails, the system lock-up is no longer monitored by the WDT because the RSWDT performs the monitoring. Thus, there is no lack of safety irrespective of the external operating conditions. The RSWDT shares the same features as the WDT (i.e., a 12-bit down counter that allows a watchdog period of up to 16 seconds with slow clock at 32.768 kHz). It can generate a general reset or a processor reset only. In addition, it can be stopped while the processor is in debug mode or idle mode.

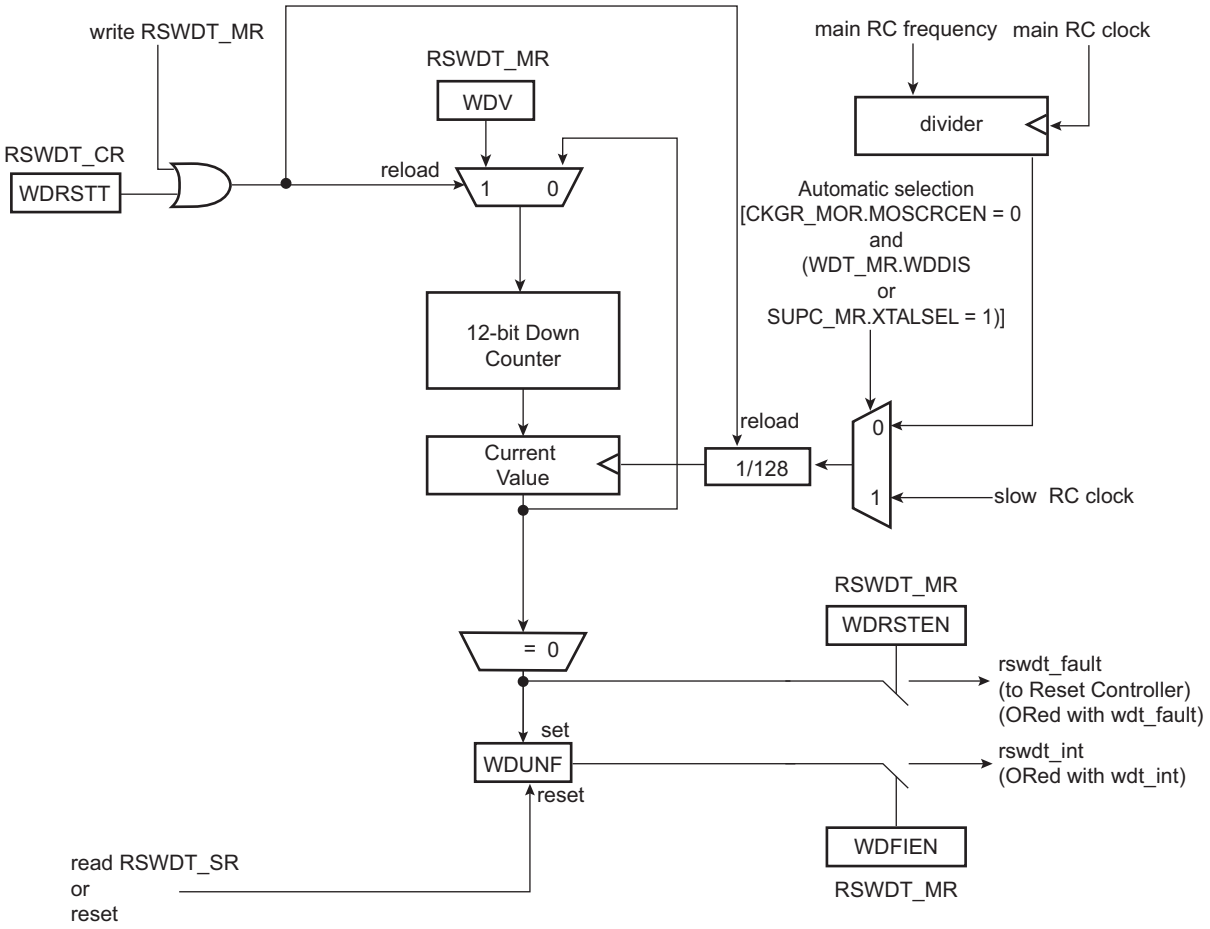
24.2 Embedded Characteristics

- Automatically Selected Reliable RSWDT Clock Source (independent of WDT clock source)
- 12-bit Key-protected Programmable Counter
- Provides Reset or Interrupt Signals to the System
- Counter may be Stopped While Processor is in Debug State or Idle Mode

Downloaded from Arrow.com.

24.3 Block Diagram

Figure 24-1. Reinforced Safety Watchdog Timer Block Diagram



Downloaded from Arrow.com.

24.4 Functional Description

The RSWDT is supplied by VDDCORE. The RSWDT is initialized with default values on processor reset or on a power-on sequence and is disabled (its default mode) under such conditions.

The RSWDT must not be enabled if the WDT is disabled.

The main RC oscillator divided clock is selected if the main RC oscillator is already enabled by the application (CKGR_MOR.MOSCRCEN = 1) or if the WDT is driven by the slow RC oscillator.

The RSWDT is built around a 12-bit down counter, which is loaded with a slow clock value other than that of the slow clock in the WDT, defined in the WDV (Watchdog Counter Value) field of the Mode Register (RSWDT_MR). The RSWDT uses the slow clock divided by 128 to establish the maximum watchdog period to be 16 seconds (with a typical slow clock of 32.768 kHz).

After a processor reset, the value of WDV is 0xFFF, corresponding to the maximum value of the counter with the external reset generation enabled (RSWDT_MR.WDRSTEN = 1 after a backup reset). This means that a default watchdog is running at reset, i.e., at power-up.

If the watchdog is restarted by writing into the Control Register (RSWDT_CR), the RSWDT_MR must not be programmed during a period of time of three slow clock periods following the RSWDT_CR write access. Programming a new value in the RSWDT_MR automatically initiates a restart instruction.

RSWDT_MR can be written only once. Only a processor reset resets it. Writing RSWDT_MR reloads the timer with the newly programmed mode parameters.

In normal operation, the user reloads the watchdog at regular intervals before the timer underflow occurs, by setting bit RSWDT_CR.WDRSTT. The watchdog counter is then immediately reloaded from the RSWDT_MR and restarted, and the slow clock 128 divider is reset and restarted. The RSWDT_CR is write-protected. As a result, writing RSWDT_CR without the correct hard-coded key has no effect. If an underflow does occur, the “wdt_fault” signal to the reset controller is asserted if the bit RSWDT_MR.WDRSTEN is set. Moreover, the bit WDUNF (Watchdog Underflow) is set in the Status Register (RSWDT_SR).

The status bits WDUNF and WDERR trigger an interrupt, provided the WDFIEN bit is set in the RSWDT_MR. The signal “wdt_fault” to the reset controller causes a Watchdog reset if the WDRSTEN bit is set as explained in the “Reset Controller (RSTC)” section of the product datasheet. In this case, the processor and the RSWDT are reset, and the WDUNF and WDERR flags are reset.

If a reset is generated, or if RSWDT_SR is read, the status bits are reset, the interrupt is cleared, and the “wdt_fault” signal to the reset controller is deasserted.

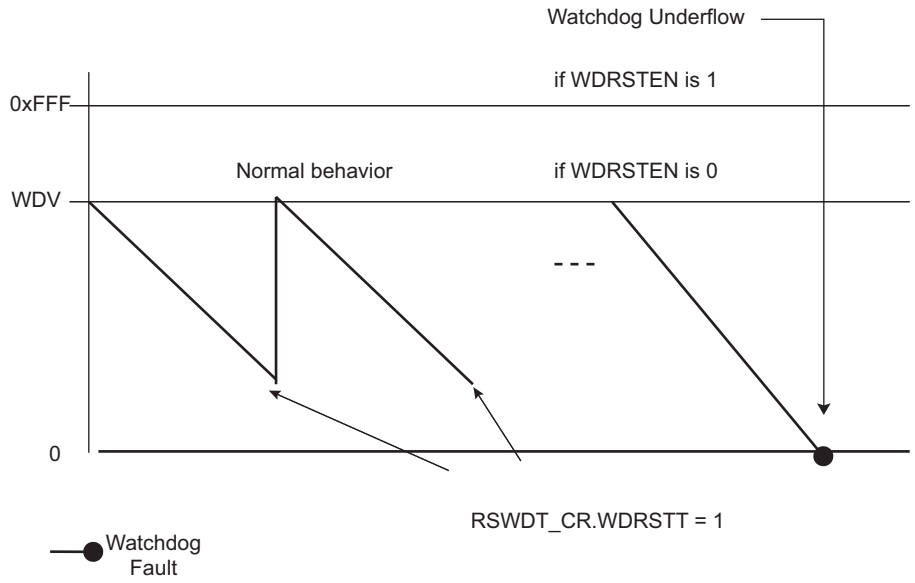
Writing RSWDT_MR reloads and restarts the down counter.

The RSWDT is disabled after any power-on sequence.

While the processor is in debug state or in idle mode, the counter may be stopped depending on the value programmed for the WDIDLEHLT and WDDBGHLT bits in the RSWDT_MR.

Downloaded from Arrow.com.

Figure 24-2. Watchdog Behavior



Downloaded from Arrow.com.

24.5 Reinforced Safety Watchdog Timer (RSWDT) User Interface

Table 24-1. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	RSWDT_CR	Write-only	–
0x04	Mode Register	RSWDT_MR	Read-write Once	0x3FFF_AFFF
0x08	Status Register	RSWDT_SR	Read-only	0x0000_0000

Downloaded from Arrow.com.

24.5.1 Reinforced Safety Watchdog Timer Control Register

Name: RSWDT_CR

Address: 0x400E1900

Access: Write-only

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WDRSTT

- **WDRSTT: Watchdog Restart**

0: No effect.

1: Restarts the watchdog.

- **KEY: Password**

Value	Name	Description
0xC4	PASSWD	Writing any other value in this field aborts the write operation.

Downloaded from Arrow.com.

24.5.2 Reinforced Safety Watchdog Timer Mode Register

Name: RSWDT_MR

Address: 0x400E1904

Access: Read-write Once

31	30	29	28	27	26	25	24
–	–	WDIDLEHLT	WDDBGHLT	ALLONES			

23	22	21	20	19	18	17	16
ALLONES							

15	14	13	12	11	10	9	8
WDDIS	–	WDRSTEN	WDFIEN	WDV			

7	6	5	4	3	2	1	0
WDV							

Note: The first write access prevents any further modification of the value of this register; read accesses remain possible.

Note: The WDV value must not be modified within three slow clock periods following a restart of the watchdog performed by means of a write access in the RSWDT_CR, else the watchdog may trigger an end of period earlier than expected.

- **WDV: Watchdog Counter Value**

Defines the value loaded in the 12-bit watchdog counter.

- **WDFIEN: Watchdog Fault Interrupt Enable**

0: A Watchdog fault (underflow or error) has no effect on interrupt.

1: A Watchdog fault (underflow or error) asserts interrupt.

- **WDRSTEN: Watchdog Reset Enable**

0: A Watchdog fault (underflow or error) has no effect on the resets.

1: A Watchdog fault (underflow or error) triggers a watchdog reset.

- **WDDIS: Watchdog Disable**

0: Enables the RSWDT.

1: Disables the RSWDT.

- **ALLONES: Must Always Be Written with 0xFFF**

- **WDDBGHLT: Watchdog Debug Halt**

0: The RSWDT runs when the processor is in debug state.

1: The RSWDT stops when the processor is in debug state.

- **WDIDLEHLT: Watchdog Idle Halt**

0: The RSWDT runs when the system is in idle mode.

1: The RSWDT stops when the system is in idle state.

Downloaded from Arrow.com.

24.5.3 Reinforced Safety Watchdog Timer Status Register

Name: RSWDT_SR

Address: 0x400E1908

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WDUNF

- **WDUNF: Watchdog Underflow**

0: No watchdog underflow occurred since the last read of RSWDT_SR.

1: At least one watchdog underflow occurred since the last read of RSWDT_SR.

Downloaded from Arrow.com.

25. Reset Controller (RSTC)

25.1 Description

The Reset Controller (RSTC), based on power-on reset cells, handles all the resets of the system without any external components. It reports which reset occurred last.

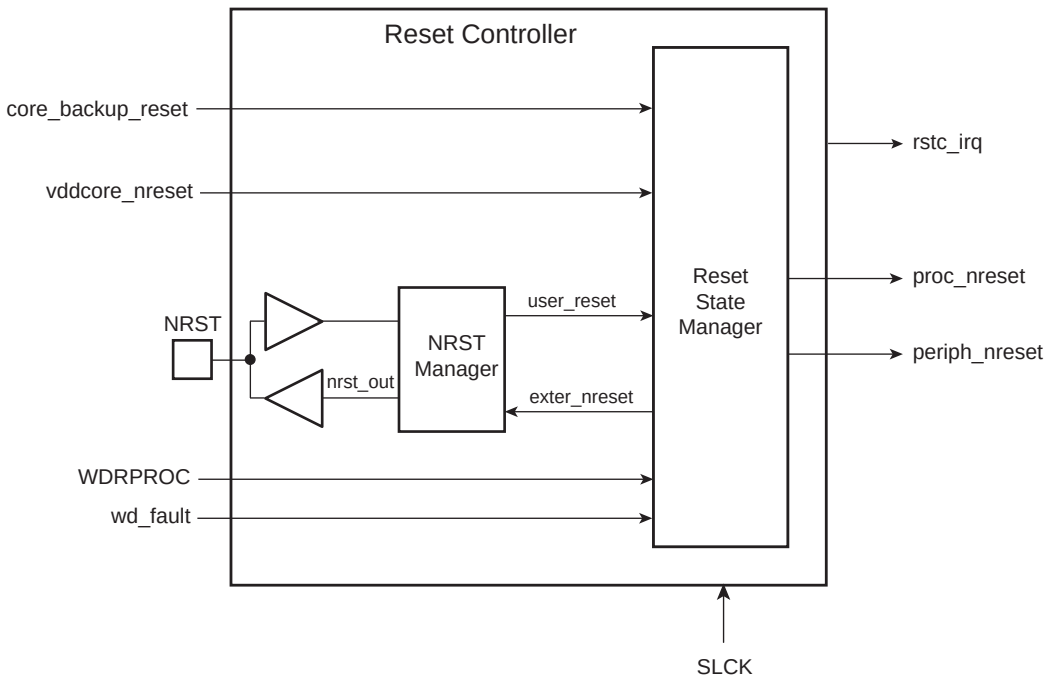
The Reset Controller also drives independently or simultaneously the external reset and the peripheral and processor resets.

25.2 Embedded Characteristics

- Management of All System Resets, Including
  - External Devices through the NRST Pin
  - Processor Reset
- Based on Embedded Power-on Cell
- Reset Source Status
  - Status of the Last Reset
  - Either Software Reset, User Reset, Watchdog Reset
- External Reset Signal Shaping

25.3 Block Diagram

Figure 25-1. Reset Controller Block Diagram



Downloaded from Arrow.com.

25.4 Functional Description

25.4.1 Reset Controller Overview

The Reset Controller is made up of an NRST manager and a reset state manager. It runs at slow clock and generates the following reset signals:

- proc_nreset: processor reset line (also resets the Watchdog Timer)
- periph_nreset: affects the whole set of embedded peripherals
- nrst_out: drives the NRST pin

These reset signals are asserted by the Reset Controller, either on events generated by peripherals, events on NRST pin, or on software action. The reset state manager controls the generation of reset signals and provides a signal to the NRST manager when an assertion of the NRST pin is required.

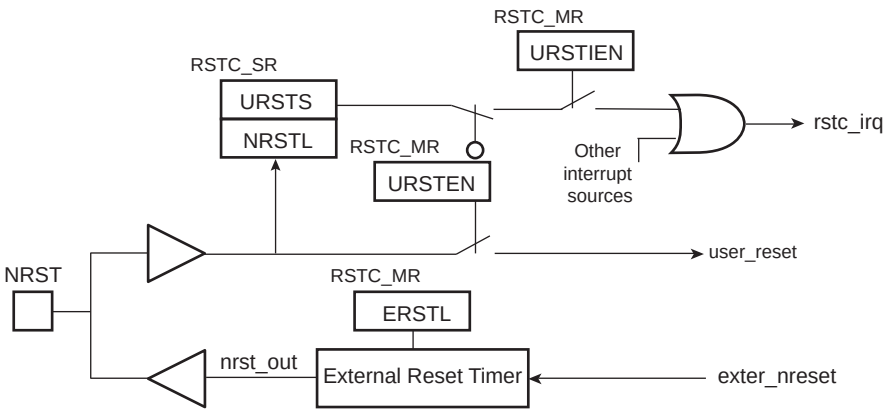
The NRST manager shapes the NRST assertion during a programmable time, thus controlling external device resets.

The Reset Controller Mode Register (RSTC_MR), used to configure the Reset Controller, is powered with VDDIO, so that its configuration is saved as long as VDDIO is on.

25.4.2 NRST Manager

The NRST manager samples the NRST input pin and drives this pin low when required by the reset state manager. Figure 25-2 shows the block diagram of the NRST manager.

Figure 25-2. NRST Manager



25.4.2.1 NRST Signal or Interrupt

The NRST manager samples the NRST pin at slow clock speed. When the line is detected low, a User Reset is reported to the reset state manager.

However, the NRST manager can be programmed to not trigger a reset when an assertion of NRST occurs. Writing a 0 to the URSTEN bit in the RSTC_MR disables the User Reset trigger.

The level of the pin NRST can be read at any time in the bit NRSTL (NRST level) in the Reset Controller Status Register (RSTC_SR). As soon as the NRST pin is asserted, bit URSTS in the RSTC_SR is set. This bit is cleared only when the RSTC_SR is read.

The Reset Controller can also be programmed to generate an interrupt instead of generating a reset. To do so, set the URSTIEN bit in the RSTC_MR.

Downloaded from Arrow.com.

25.4.2.2 NRST External Reset Control

The reset state manager asserts the signal exter_nreset to assert the NRST pin. When this occurs, the “nrst_out” signal is driven low by the NRST manager for a time programmed by field ERSTL in the RSTC_MR. This assertion duration, named External Reset Length, lasts $2^{(ERSTL+1)}$ slow clock cycles. This gives the approximate duration of an assertion between 60 µs and 2 seconds. Note that ERSTL at 0 defines a two-cycle duration for the NRST pulse.

This feature allows the Reset Controller to shape the NRST pin level, and thus to guarantee that the NRST line is driven low for a time compliant with potential external devices connected on the system reset.

RSTC_MR is backed up, making it possible to use the ERSTL field to shape the system power-up reset for devices requiring a longer startup time than that of the slow clock oscillator.

25.4.3 Reset States

The reset state manager handles the different reset sources and generates the internal reset signals. It reports the reset status in field RSTTYP of the Status Register (RSTC_SR). The update of RSTC_SR.RSTTYP is performed when the processor reset is released.

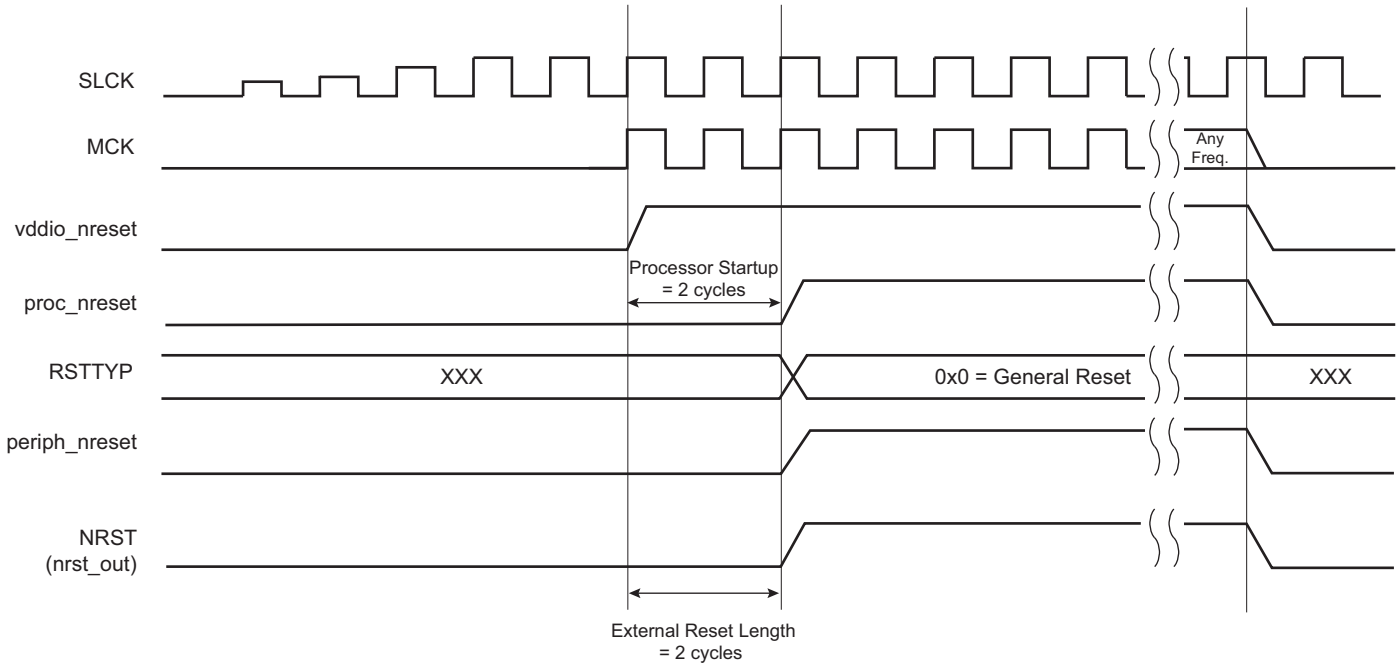
25.4.3.1 General Reset

A general reset occurs when a VDDIO power-on-reset is detected, a brownout or a voltage regulation loss is detected by the Supply Controller. The vddcore_nreset signal is asserted by the Supply Controller when a general reset occurs.

All the reset signals are released and field RSTC_SR.RSTTYP reports a general reset. As the RSTC_MR is reset, the NRST line rises two cycles after the vddcore_nreset, as ERSTL defaults at value 0x0.

Figure 25-3 shows how the general reset affects the reset signals.

Figure 25-3. General Reset State



Downloaded from Arrow.com.

25.4.3.2 Backup Reset

A backup reset occurs when the chip exits from Backup mode. While exiting Backup mode, the vddcore_nreset signal is asserted by the Supply Controller.

Field RSTC_SR.RSTTYP is updated to report a backup reset.

25.4.3.3 Watchdog Reset

The watchdog reset is entered when a watchdog fault occurs. This reset lasts three slow clock cycles.

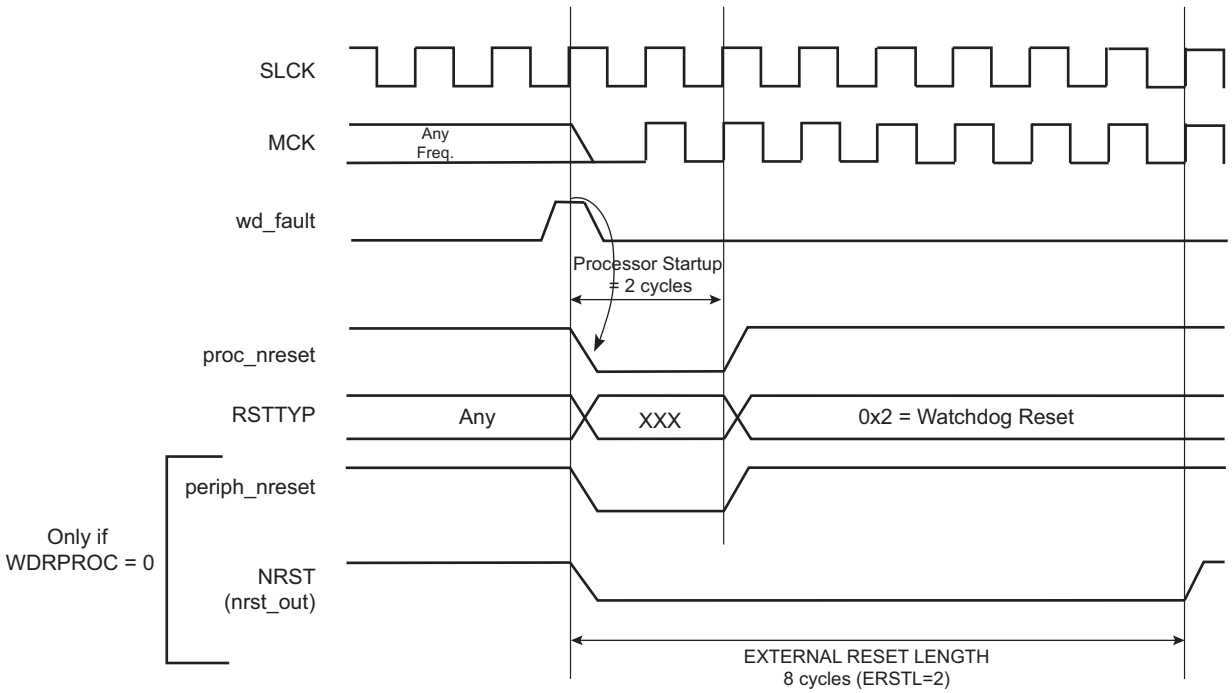
When in watchdog reset, assertion of the reset signals depends on the WDRPROC bit in the WDT_MR:

- If WDRPROC = 0, the processor reset and the peripheral reset are asserted. The NRST line is also asserted, depending on how field RSTC_MR.ERSTL is programmed. However, the resulting low level on NRST does not result in a user reset state.
- If WDRPROC = 1, only the processor reset is asserted.

The Watchdog Timer is reset by the proc_nreset signal. As the watchdog fault always causes a processor reset if WDRSTEN in the WDT_MR is set, the Watchdog Timer is always reset after a watchdog reset, and the Watchdog is enabled by default and with a period set to a maximum.

When bit WDT_MR.WDRSTEN is reset, the watchdog fault has no impact on the Reset Controller.

Figure 25-4. Watchdog Reset



Downloaded from Arrow.com.

25.4.3.4 Software Reset

The Reset Controller offers commands to assert the different reset signals. These commands are performed by writing the Control Register (RSTC_CR) with the following bits at 1:

- RSTC_CR.PROCRST: Writing a 1 to PROCRST resets the processor and all the embedded peripherals, including the memory system and, in particular, the Remap Command.
- RSTC_CR.EXTRST: Writing a 1 to EXTRST asserts low the NRST pin during a time defined by the field RSTC_MR.ERSTL.

The software reset is entered if at least one of these bits is set by the software. All these commands can be performed independently or simultaneously. The software reset lasts three slow clock cycles.

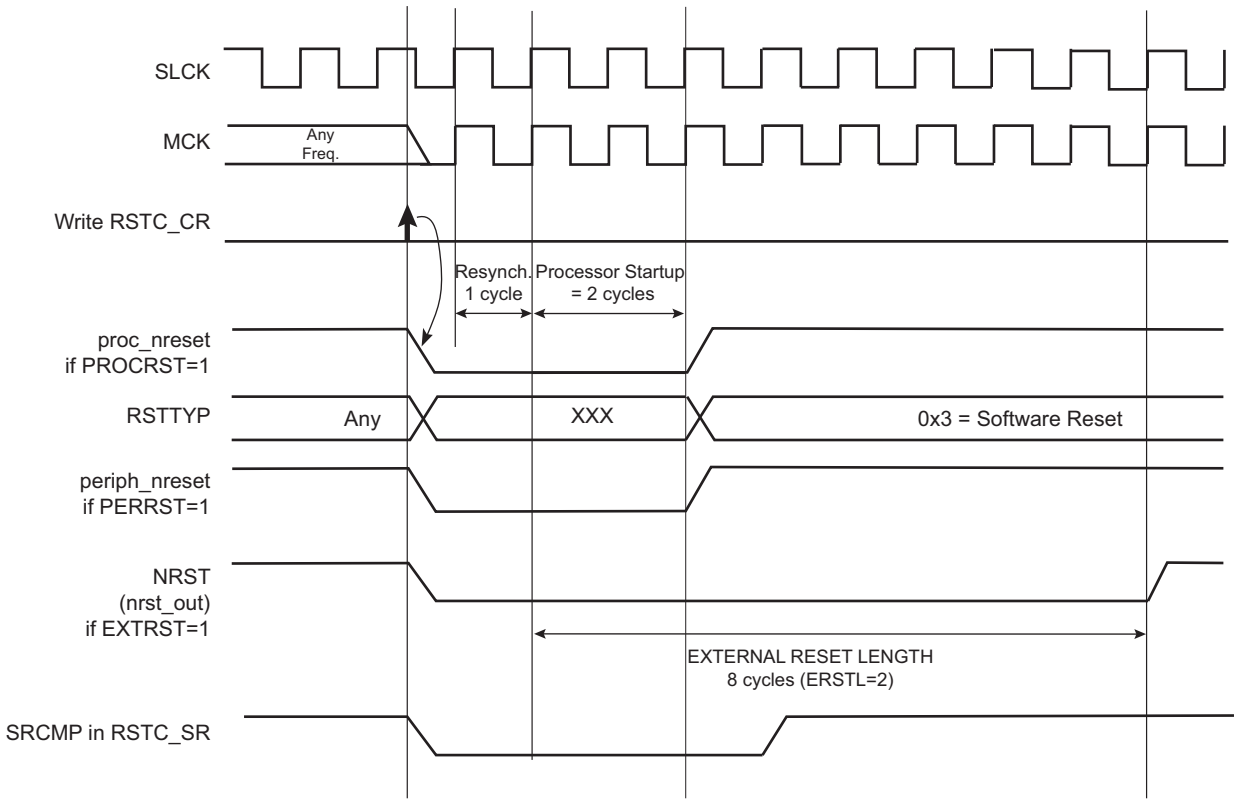
The internal reset signals are asserted as soon as the register write is performed. This is detected on the Master Clock (MCK). They are released when the software reset has ended, i.e., synchronously to SLCK.

If EXTRST is set, the nrst_out signal is asserted depending on the configuration of field RSTC_MR.ERSTL. However, the resulting falling edge on NRST does not lead to a user reset.

If and only if the PROCRST bit is set, the Reset Controller reports the software status in field RSTC_SR.RSTTYP. Other software resets are not reported in RSTTYP.

As soon as a software operation is detected, the bit SRCMP (Software Reset Command in Progress) is set in the RSTC_SR. SRCMP is cleared at the end of the software reset. No other software reset can be performed while the SRCMP bit is set, and writing any value in the RSTC_CR has no effect.

Figure 25-5. Software Reset



Downloaded from Arrow.com.

25.4.3.5 User Reset

The user reset is entered when a low level is detected on the NRST pin and bit URSTEN in the RSTC_MR is at 1. The NRST input signal is resynchronized with SLCK to insure proper behavior of the system.

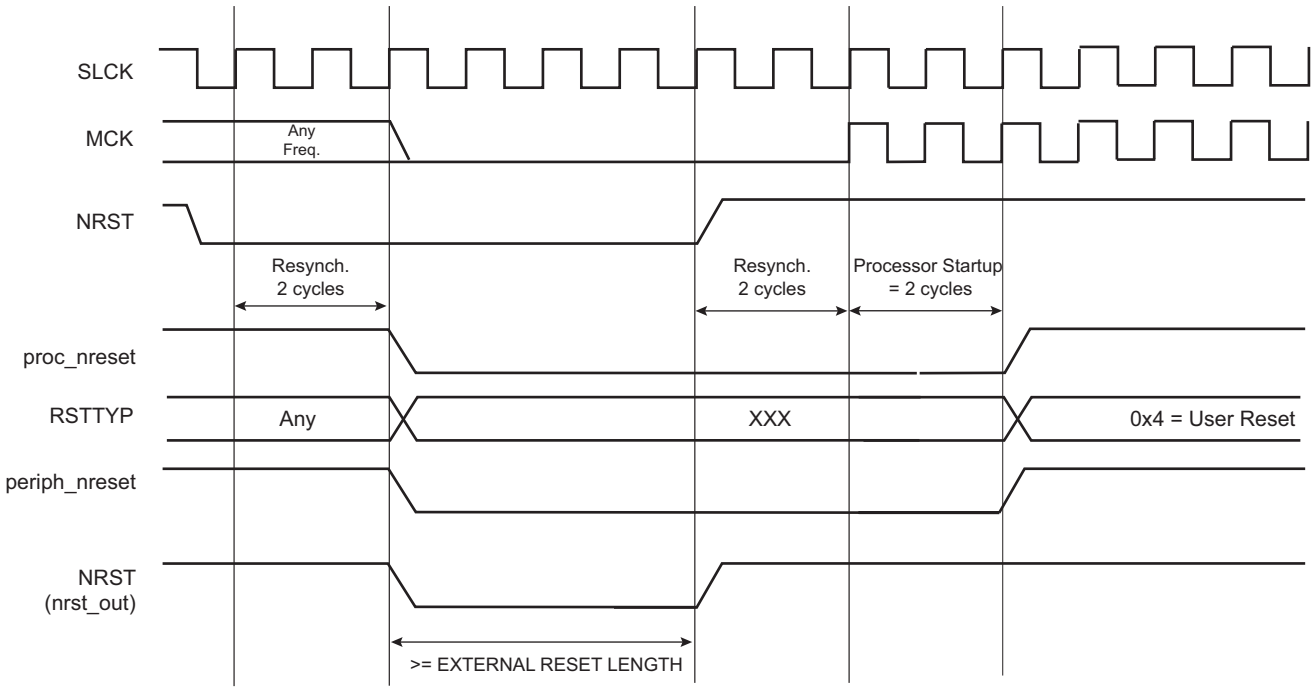
The user reset is entered as soon as a low level is detected on NRST. The processor reset and the peripheral reset are asserted.

The user reset ends when NRST rises, after a two-cycle resynchronization time and a three-cycle processor startup. The processor clock is re-enabled as soon as NRST is confirmed high.

When the processor reset signal is released, field RSTC_SR.RSTTYP is loaded with the value 0x4, indicating a user reset.

The NRST manager guarantees that the NRST line is asserted for External Reset Length slow clock cycles, as programmed in field RSTC_MR.ERSTL. However, if NRST does not rise after External Reset Length because it is driven low externally, the internal reset lines remain asserted until NRST actually rises.

Figure 25-6. User Reset State



Downloaded from Arrow.com.

25.4.4 Reset State Priorities

The reset state manager manages the priorities among the different reset sources. The resets are listed in order of priority as follows:

1. General reset
2. Backup reset
3. Watchdog reset
4. Software reset
5. User reset

Particular cases are listed below:

- When in user reset:
  - A watchdog event is impossible because the Watchdog Timer is being reset by the proc_nreset signal.
  - A software reset is impossible, since the processor reset is being activated.
- When in software reset:
  - A watchdog event has priority over the current state.
  - The NRST has no effect.
- When in watchdog reset:
  - The processor reset is active and so a software reset cannot be programmed.
  - A user reset cannot be entered.

Downloaded from Arrow.com.

25.5 Reset Controller (RSTC) User Interface

Table 25-1. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	RSTC_CR	Write-only	–
0x04	Status Register	RSTC_SR	Read-only	0x0000_0000[1]
0x08	Mode Register	RSTC_MR	Read/Write	0x0000_0001

Note: 1. This value assumes that a general reset has been performed, subject to change if other types of reset are generated.

Downloaded from Arrow.com.

25.5.1 Reset Controller Control Register

Name: RSTC_CR

Address: 0x400E1800

Access: Write-only

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	EXTRST	–	–	PROCRST

- **PROCRST: Processor Reset**

0: No effect

1: If KEY is correct, resets the processor

- **EXTRST: External Reset**

0: No effect

1: If KEY is correct, asserts the NRST pin

- **KEY: System Reset Key**

Value	Name	Description
0xA5	PASSWD	Writing any other value in this field aborts the write operation.

Downloaded from Arrow.com.

25.5.2 Reset Controller Status Register

Name: RSTC_SR

Address: 0x400E1804

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	SRCMP	NRSTL
15	14	13	12	11	10	9	8
–	–	–	–	–	RSTTYP		
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	URSTS

- **URSTS: User Reset Status**

A high-to-low transition of the NRST pin sets the URSTS bit. This transition is also detected on the MCK rising edge. If the user reset is disabled (URSTEN = 0 in RSTC_MR) and if the interruption is enabled by the URSTIEN bit in the RSTC_MR, the URSTS bit triggers an interrupt. Reading the RSTC_SR resets the URSTS bit and clears the interrupt.

0: No high-to-low edge on NRST happened since the last read of RSTC_SR.

1: At least one high-to-low transition of NRST has been detected since the last read of RSTC_SR.

- **RSTTYP: Reset Type**

This field reports the cause of the last processor reset. Reading this RSTC_SR does not reset this field.

Value	Name	Description
0	GENERAL_RST	First power-up reset
1	BACKUP_RST	Return from Backup Mode
2	WDT_RST	Watchdog fault occurred
3	SOFT_RST	Processor reset required by the software
4	USER_RST	NRST pin detected low
5	–	Reserved
6	–	Reserved
7	–	Reserved

- **NRSTL: NRST Pin Level**

This bit registers the NRST pin level sampled on each Master Clock (MCK) rising edge.

- **SRCMP: Software Reset Command in Progress**

When set, this bit indicates that a software reset command is in progress and that no further software reset should be performed until the end of the current one. This bit is automatically cleared at the end of the current software reset.

0: No software command is being performed by the Reset Controller. The Reset Controller is ready for a software command.

1: A software reset command is being performed by the Reset Controller. The Reset Controller is busy.

Downloaded from Arrow.com.

25.5.3 Reset Controller Mode Register

Name: RSTC_MR

Address: 0x400E1808

Access: Read/Write

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	ERSTL			

7	6	5	4	3	2	1	0
–	–	–	URSTIEN	–	–	–	URSTEN

This register can only be written if the WPEN bit is cleared in the System Controller Write Protection Mode Register (SYSC_WPMR).

- **URSTEN: User Reset Enable**

0: The detection of a low level on the NRST pin does not generate a user reset.

1: The detection of a low level on the NRST pin triggers a user reset.

- **URSTIEN: User Reset Interrupt Enable**

0: USRTS bit in RSTC_SR at 1 has no effect on rstc_irq.

1: USRTS bit in RSTC_SR at 1 asserts rstc_irq if URSTEN = 0.

- **ERSTL: External Reset Length**

This field defines the external reset length. The external reset is asserted during a time of $2^{(ERSTL+1)}$ slow clock cycles. This allows assertion duration to be programmed between 60 µs and 2 seconds. Note that synchronization cycles must also be considered when calculating the actual reset length as previously described.

- **KEY: Write Access Password**

Value	Name	Description
0xA5	PASSWD	Writing any other value in this field aborts the write operation. Always reads as 0.

Downloaded from Arrow.com.

26. Real-time Clock (RTC)

26.1 Description

The Real-time Clock (RTC) peripheral is designed for very low power consumption. For optimal functionality, the RTC requires an accurate external 32.768 kHz clock, which can be provided by a crystal oscillator.

It combines a complete time-of-day clock with alarm and a Gregorian or Persian calendar, complemented by a programmable periodic interrupt. The alarm and calendar registers are accessed by a 32-bit data bus.

The time and calendar values are coded in binary-coded decimal (BCD) format. The time format can be 24-hour mode or 12-hour mode with an AM/PM indicator.

Updating time and calendar fields and configuring the alarm fields are performed by a parallel capture on the 32-bit data bus. An entry control is performed to avoid loading registers with incompatible BCD format data or with an incompatible date according to the current month/year/century.

A clock divider calibration circuitry can be used to compensate for crystal oscillator frequency variations.

An RTC output can be programmed to generate several waveforms, including a prescaled clock derived from 32.768 kHz.

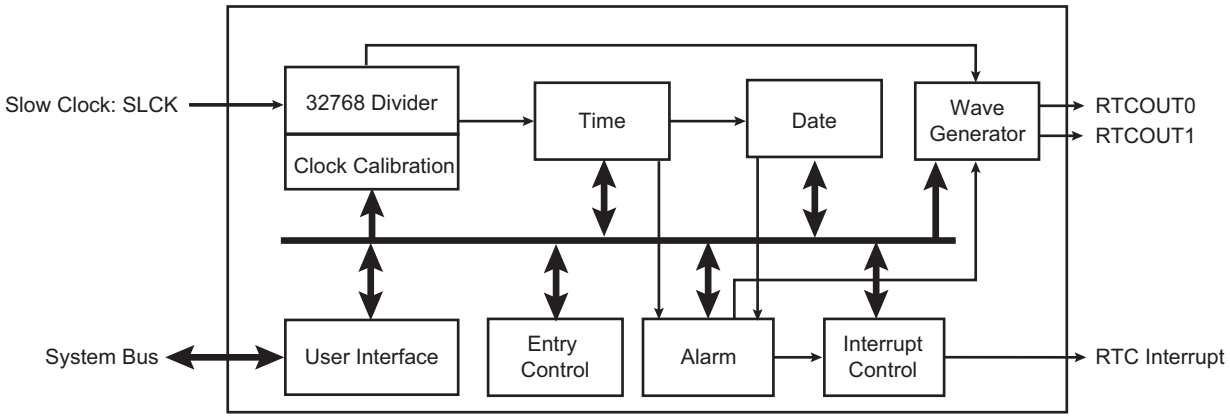
26.2 Embedded Characteristics

- Full Asynchronous Design for Ultra Low Power Consumption
- Gregorian and Persian Modes Supported
- Programmable Periodic Interrupt
- Safety/security Features:
  - Valid Time and Date Programmation Check
  - On-The-Fly Time and Date Validity Check
- Counters Calibration Circuitry to Compensate for Crystal Oscillator Variations
- Waveform Generation
- Register Write Protection

Downloaded from Arrow.com.

26.3 Block Diagram

Figure 26-1. Real-time Clock Block Diagram



26.4 Product Dependencies

26.4.1 Power Management

The Real-time Clock is continuously clocked at 32.768 kHz. The Power Management Controller has no effect on RTC behavior.

26.4.2 Interrupt

Within the System Controller, the RTC interrupt is OR-wired with all the other module interrupts.

Only one System Controller interrupt line is connected on one of the internal sources of the interrupt controller.

RTC interrupt requires the interrupt controller to be programmed first.

When a System Controller interrupt occurs, the service routine must first determine the cause of the interrupt. This is done by reading each status register of the System Controller peripherals successively.

Table 26-1. Peripheral IDs

Instance	ID
RTC	2

Downloaded from Arrow.com.

26.5 Functional Description

The RTC provides a full binary-coded decimal (BCD) clock that includes century (19/20), year (with leap years), month, date, day, hours, minutes and seconds reported in RTC Time Register (RTC_TIMR) and RTC Calendar Register (RTC_CALR).

The valid year range is up to 2099 in Gregorian mode (or 1300 to 1499 in Persian mode).

The RTC can operate in 24-hour mode or in 12-hour mode with an AM/PM indicator.

Corrections for leap years are included (all years divisible by 4 being leap years except 1900). This is correct up to the year 2099.

The RTC can generate configurable waveforms on RTCOUT0/1 outputs.

26.5.1 Reference Clock

The reference clock is the Slow Clock (SLCK). It can be driven internally or by an external 32.768 kHz crystal.

During low power modes of the processor, the oscillator runs and power consumption is critical. The crystal selection has to take into account the current consumption for power saving and the frequency drift due to temperature effect on the circuit for time accuracy.

26.5.2 Timing

The RTC is updated in real time at one-second intervals in Normal mode for the counters of seconds, at one-minute intervals for the counter of minutes and so on.

Due to the asynchronous operation of the RTC with respect to the rest of the chip, to be certain that the value read in the RTC registers (century, year, month, date, day, hours, minutes, seconds) are valid and stable, it is necessary to read these registers twice. If the data is the same both times, then it is valid. Therefore, a minimum of two and a maximum of three accesses are required.

26.5.3 Alarm

The RTC has five programmable fields: month, date, hours, minutes and seconds.

Each of these fields can be enabled or disabled to match the alarm condition:

- If all the fields are enabled, an alarm flag is generated (the corresponding flag is asserted and an interrupt generated if enabled) at a given month, date, hour/minute/second.
- If only the “seconds” field is enabled, then an alarm is generated every minute.

Depending on the combination of fields enabled, a large number of possibilities are available to the user ranging from minutes to 365/366 days.

Hour, minute and second matching alarm (SECEN, MINEN, HOUREN) can be enabled independently of SEC, MIN, HOUR fields.

Note: To change one of the SEC, MIN, HOUR, DATE, MONTH fields, it is recommended to disable the field before changing the value and then re-enable it after the change has been made. This requires up to three accesses to the RTC_TIMALR or RTC_CALALR. The first access clears the enable corresponding to the field to change (SECEN, MINEN, HOUREN, DATEEN, MTHEN). If the field is already cleared, this access is not required. The second access performs the change of the value (SEC, MIN, HOUR, DATE, MONTH). The third access is required to re-enable the field by writing 1 in SECEN, MINEN, HOUREn, DATEEN, MTHEN fields.

26.5.4 Error Checking when Programming

Verification on user interface data is performed when accessing the century, year, month, date, day, hours, minutes, seconds and alarms. A check is performed on illegal BCD entries such as illegal date of the month with regard to the year and century configured.

Downloaded from Arrow.com.

If one of the time fields is not correct, the data is not loaded into the register/counter and a flag is set in the validity register. The user can not reset this flag. It is reset as soon as an acceptable value is programmed. This avoids any further side effects in the hardware. The same procedure is followed for the alarm.

The following checks are performed:

1. Century (check if it is in range 19–20 or 13–14 in Persian mode)
2. Year (BCD entry check)
3. Date (check range 01–31)
4. Month (check if it is in BCD range 01–12, check validity regarding “date”)
5. Day (check range 1–7)
6. Hour (BCD checks: in 24-hour mode, check range 00–23 and check that AM/PM flag is not set if RTC is set in 24-hour mode; in 12-hour mode check range 01–12)
7. Minute (check BCD and range 00–59)
8. Second (check BCD and range 00–59)

Note: If the 12-hour mode is selected by means of the RTC Mode Register (RTC_MR), a 12-hour value can be programmed and the returned value on RTC_TIMR will be the corresponding 24-hour value. The entry control checks the value of the AM/PM indicator (bit 22 of RTC_TIMR) to determine the range to be checked.

26.5.5 RTC Internal Free Running Counter Error Checking

To improve the reliability and security of the RTC, a permanent check is performed on the internal free running counters to report non-BCD or invalid date/time values.

An error is reported by TDERR bit in the status register (RTC_SR) if an incorrect value has been detected. The flag can be cleared by setting the TDERRCLR bit in the Status Clear Command Register (RTC_SCCR).

Anyway the TDERR error flag will be set again if the source of the error has not been cleared before clearing the TDERR flag. The clearing of the source of such error can be done by reprogramming a correct value on RTC_CALR and/or RTC_TIMR.

The RTC internal free running counters may automatically clear the source of TDERR due to their roll-over (i.e., every 10 seconds for SECONDS[3:0] field in RTC_TIMR). In this case the TDERR is held high until a clear command is asserted by TDERRCLR bit in RTC_SCCR.

26.5.6 Updating Time/Calendar

To update any of the time/calendar fields, the user must first stop the RTC by setting the corresponding field in the Control Register (RTC_CR). Bit UPDTIM must be set to update time fields (hour, minute, second) and bit UPDCAL must be set to update calendar fields (century, year, month, date, day).

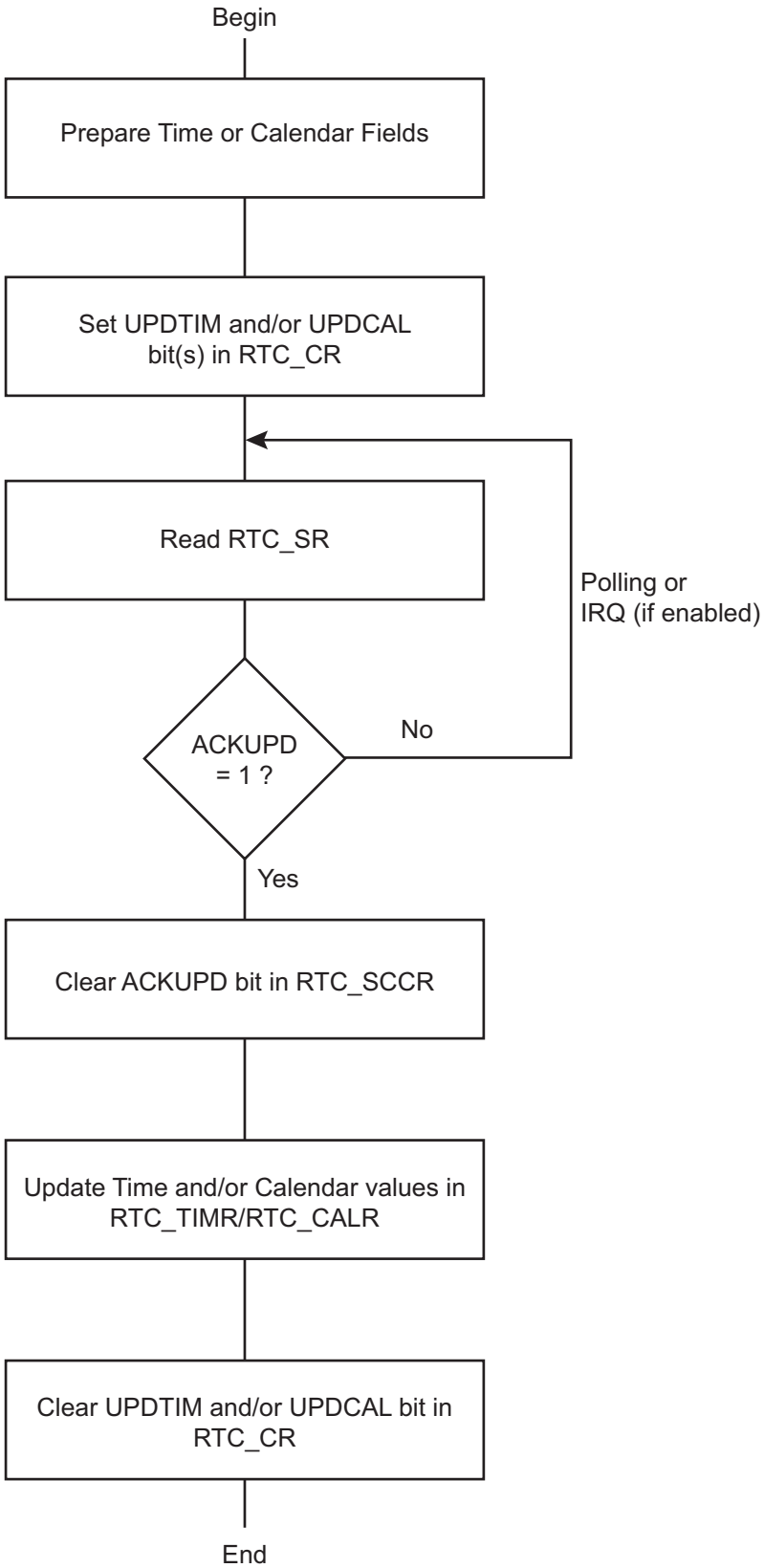
The ACKUPD bit is automatically set within a second after setting the UPDTIM and/or UPDCAL bit (meaning one second is the maximum duration of the polling or wait for interrupt period). Once ACKUPD is set, it is mandatory to clear this flag by writing the corresponding bit in the RTC_SCCR, after which the user can write to the Time Register, the Calendar Register, or both.

Once the update is finished, the user must clear UPDTIM and/or UPDCAL in the RTC_CR.

When entering the programming mode of the calendar fields, the time fields remain enabled. When entering the programming mode of the time fields, both time and calendar fields are stopped. This is due to the location of the calendar logic circuity (downstream for low-power considerations). It is highly recommended to prepare all the fields to be updated before entering programming mode. In successive update operations, the user must wait at least one second after resetting the UPDTIM/UPDCAL bit in the RTC_CR before setting these bits again. This is done by waiting for the SEC flag in the RTC_SR before setting UPDTIM/UPDCAL bit. After clearing UPDTIM/UPDCAL, the SEC flag must also be cleared.

Downloaded from Arrow.com.

Figure 26-2. Update Sequence



Downloaded from Arrow.com.

26.5.7 RTC Accurate Clock Calibration

The crystal oscillator that drives the RTC may not be as accurate as expected mainly due to temperature variation. The RTC is equipped with circuitry able to correct slow clock crystal drift.

To compensate for possible temperature variations over time, this accurate clock calibration circuitry can be programmed on-the-fly and also programmed during application manufacturing, in order to correct the crystal frequency accuracy at room temperature (20–25°C). The typical clock drift range at room temperature is ±20 ppm.

In the device operating temperature range, the 32.768 kHz crystal oscillator clock inaccuracy can be up to -200 ppm.

The RTC clock calibration circuitry allows positive or negative correction in a range of 1.5 ppm to 1950 ppm.

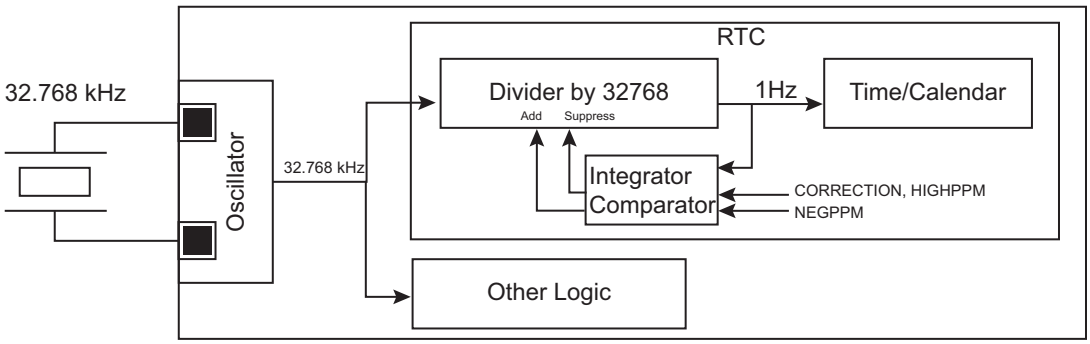
The calibration circuitry is fully digital. Thus, the configured correction is independent of temperature, voltage, process, etc., and no additional measurement is required to check that the correction is effective.

If the correction value configured in the calibration circuitry results from an accurate crystal frequency measure, the remaining accuracy is bounded by the values listed below:

- Below 1 ppm, for an initial crystal drift between 1.5 ppm up to 20 ppm, and from 30 ppm to 90 ppm
- Below 2 ppm, for an initial crystal drift between 20 ppm up to 30 ppm, and from 90 ppm to 130 ppm
- Below 5 ppm, for an initial crystal drift between 130 ppm up to 200 ppm

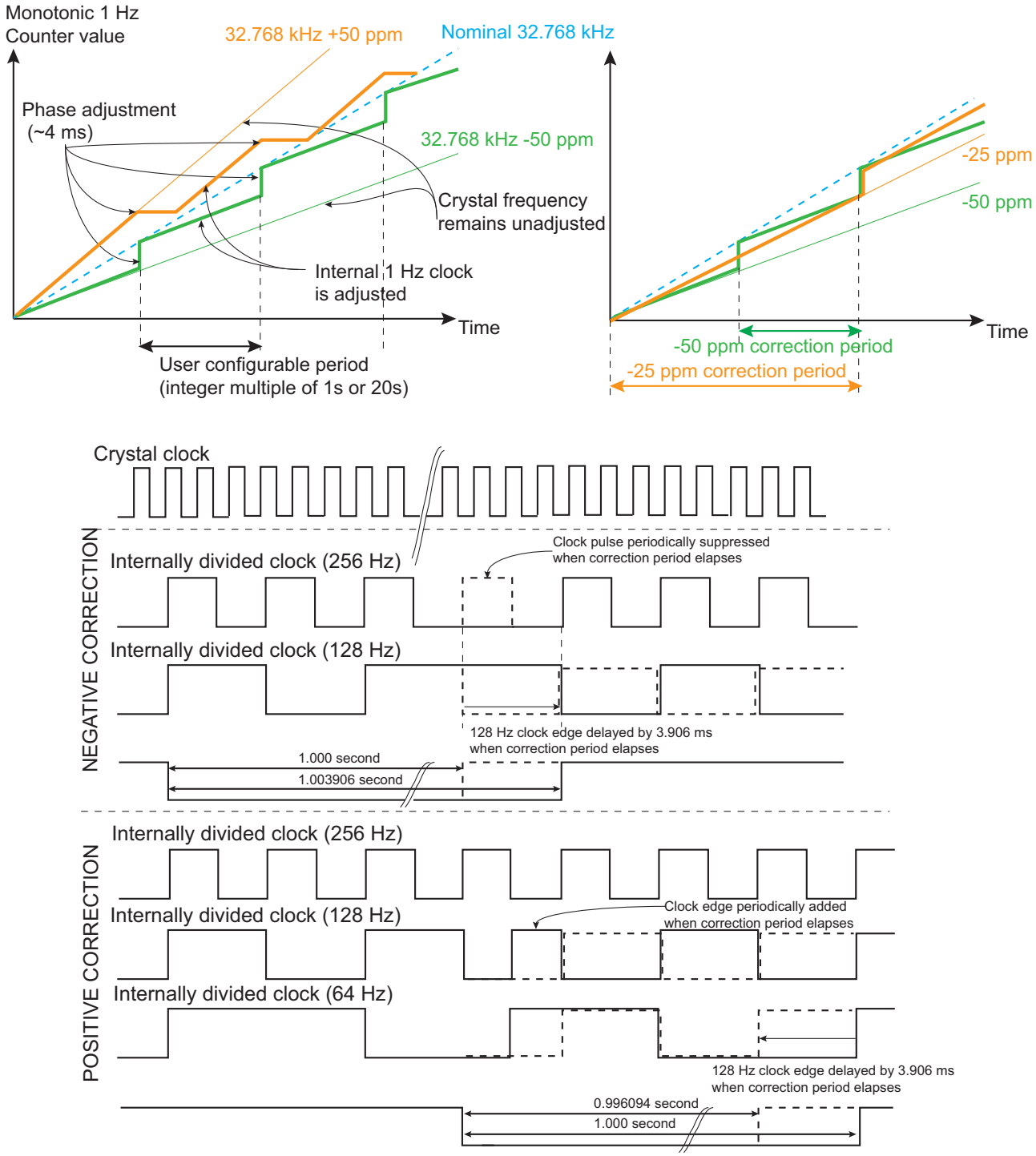
The calibration circuitry does not modify the 32.768 kHz crystal oscillator clock frequency but it acts by slightly modifying the 1 Hz clock period from time to time. The correction event occurs every 1 + [(20 - (19 x HIGHPPM)) x CORRECTION] seconds. When the period is modified, depending on the sign of the correction, the 1 Hz clock period increases or reduces by around 4 ms. Depending on the CORRECTION, NEGPPM and HIGHPPM values configured in RTC_MR, the period interval between two correction events differs.

Figure 26-3. Calibration Circuitry



Downloaded from Arrow.com.

Figure 26-4. Calibration Circuitry Waveforms



The inaccuracy of a crystal oscillator at typical room temperature (±20 ppm at 20–25 °C) can be compensated if a reference clock/signal is used to measure such inaccuracy. This kind of calibration operation can be set up during the final product manufacturing by means of measurement equipment embedding such a reference clock. The correction of value must be programmed into the (RTC_MR), and this value is kept as long as the circuitry is powered (backup area). Removing the backup power supply cancels this calibration. This room temperature calibration can be further processed by means of the networking capability of the target application.

Downloaded from Arrow.com.

To ease the comparison of the inherent crystal accuracy with the reference clock/signal during manufacturing, an internal prescaled 32.768 kHz clock derivative signal can be assigned to drive RTC output. To accommodate the measure, several clock frequencies can be selected among 1 Hz, 32 Hz, 64 Hz, 512 Hz.

The clock calibration correction drives the internal RTC counters but can also be observed in the RTC output when one of the following three frequencies 1 Hz, 32 Hz or 64 Hz is configured. The correction is not visible in the RTC output if 512 Hz frequency is configured.

In any event, this adjustment does not take into account the temperature variation.

The frequency drift (up to -200 ppm) due to temperature variation can be compensated using a reference time if the application can access such a reference. If a reference time cannot be used, a temperature sensor can be placed close to the crystal oscillator in order to get the operating temperature of the crystal oscillator. Once obtained, the temperature may be converted using a lookup table (describing the accuracy/temperature curve of the crystal oscillator used) and RTC_MR configured accordingly. The calibration can be performed on-the-fly. This adjustment method is not based on a measurement of the crystal frequency/drift and therefore can be improved by means of the networking capability of the target application.

If no crystal frequency adjustment has been done during manufacturing, it is still possible to do it. In the case where a reference time of the day can be obtained through LAN/WAN network, it is possible to calculate the drift of the application crystal oscillator by comparing the values read on RTC Time Register (RTC_TIMR) and programming the HIGHPPM and CORRECTION fields on RTC_MR according to the difference measured between the reference time and those of RTC_TIMR.

26.5.8 Waveform Generation

Waveforms can be generated by the RTC in order to take advantage of the RTC inherent prescalers while the RTC is the only powered circuitry (Low-power mode of operation, Backup mode) or in any active mode. Going into Backup or Low-power operating modes does not affect the waveform generation outputs.

The RTC outputs (RTCOUT0 and RTCOUT1) have a source driver selected among seven possibilities.

The first selection choice sticks the associated output at 0 (This is the reset value and it can be used at any time to disable the waveform generation).

Selection choices 1 to 4 respectively select 1 Hz, 32 Hz, 64 Hz and 512 Hz.

32 Hz or 64 Hz can drive, for example, a TN LCD backplane signal while 1 Hz can be used to drive a blinking character like “:” for basic time display (hour, minute) on TN LCDs.

Selection choice 5 provides a toggling signal when the RTC alarm is reached.

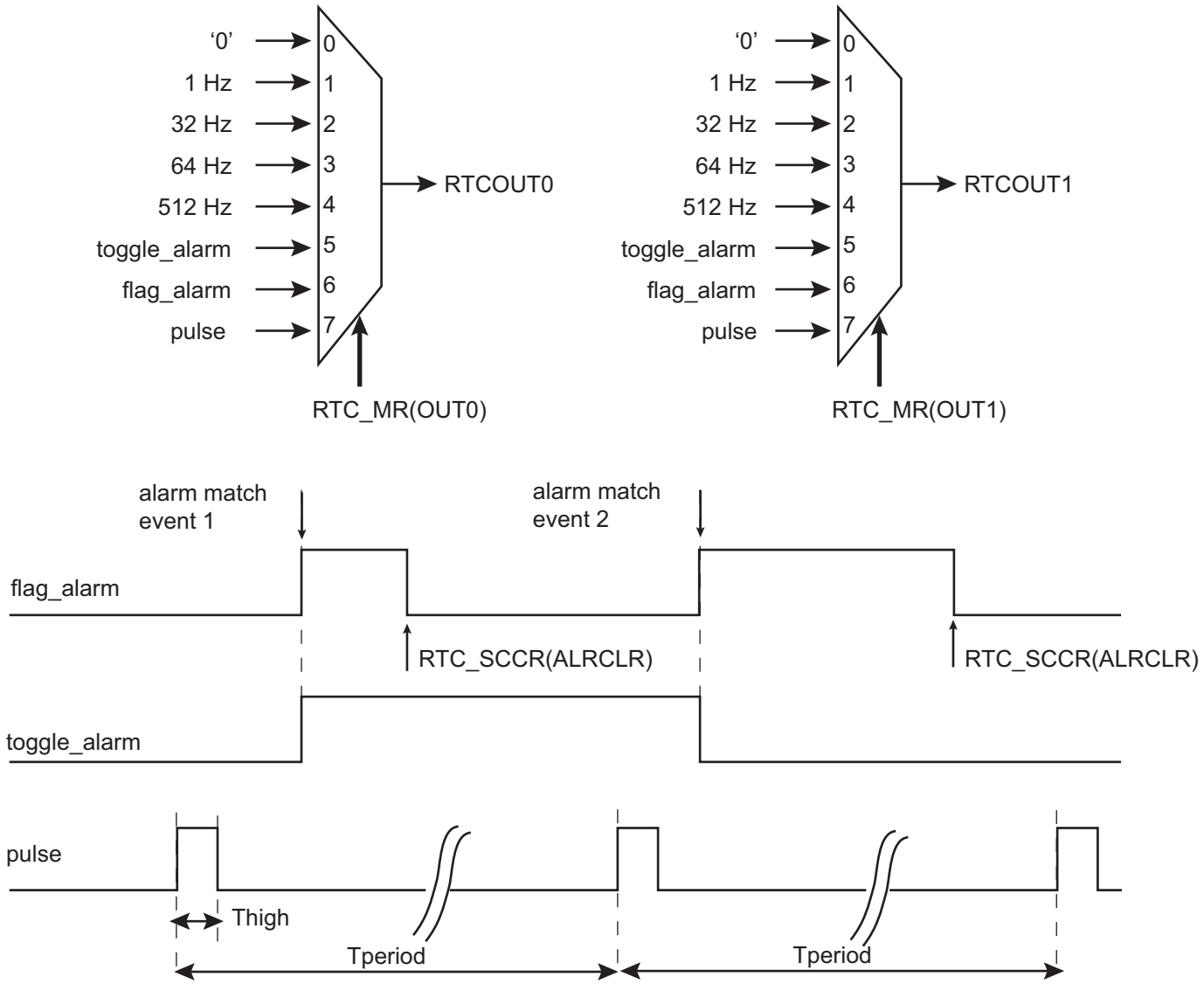
Selection choice 6 provides a copy of the alarm flag, so the associated output is set high (logical 1) when an alarm occurs and immediately cleared when software clears the alarm interrupt source.

Selection choice 7 provides a 1 Hz periodic high pulse of 15 µs duration that can be used to drive external devices for power consumption reduction or any other purpose.

PIO lines associated to RTC outputs are automatically selecting these waveforms as soon as RTC_MR corresponding fields OUT0 and OUT1 differ from 0.

Downloaded from Arrow.com.

Figure 26-5. Waveform Generation



Downloaded from Arrow.com.

26.6 Real-time Clock (RTC) User Interface

Table 26-2. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	RTC_CR	Read/Write	0x00000000
0x04	Mode Register	RTC_MR	Read/Write	0x00000000
0x08	Time Register	RTC_TIMR	Read/Write	0x00000000
0x0C	Calendar Register	RTC_CALR	Read/Write	0x01E11220
0x10	Time Alarm Register	RTC_TIMALR	Read/Write	0x00000000
0x14	Calendar Alarm Register	RTC_CALALR	Read/Write	0x01010000
0x18	Status Register	RTC_SR	Read-only	0x00000000
0x1C	Status Clear Command Register	RTC_SCCR	Write-only	–
0x20	Interrupt Enable Register	RTC_IER	Write-only	–
0x24	Interrupt Disable Register	RTC_IDR	Write-only	–
0x28	Interrupt Mask Register	RTC_IMR	Read-only	0x00000000
0x2C	Valid Entry Register	RTC_VER	Read-only	0x00000000
0x30–0xC8	Reserved	–	–	–
0xCC	Reserved	–	–	–
0xD0	Reserved	–	–	–
0xD4–0xE0	Reserved	–	–	–
0xE4	Write Protection Mode Register	RTC_WPMR	Read/Write	0x00000000
0xE8–0xF8	Reserved	–	–	–
0xFC	Reserved	–	–	–

Note: If an offset is not listed in the table it must be considered as reserved.

Downloaded from Arrow.com.

26.6.1 RTC Control Register

Name: RTC_CR

Address: 0x400E1860

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	CALEVSEL	
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TIMEVSEL	
7	6	5	4	3	2	1	0
–	–	–	–	–	–	UPDCAL	UPDTIM

This register can only be written if the WPEN bit is cleared in the RTC Write Protection Mode Register.

- **UPDTIM: Update Request Time Register**

0: No effect or, if UPDTIM has been previously written to 1, stops the update procedure.

1: Stops the RTC time counting.

Time counting consists of second, minute and hour counters. Time counters can be programmed once this bit is set and acknowledged by the bit ACKUPD of the RTC_SR.

- **UPDCAL: Update Request Calendar Register**

0: No effect or, if UPDCAL has been previously written to 1, stops the update procedure.

1: Stops the RTC calendar counting.

Calendar counting consists of day, date, month, year and century counters. Calendar counters can be programmed once this bit is set and acknowledged by the bit ACKUPD of the RTC_SR.

- **TIMEVSEL: Time Event Selection**

The event that generates the flag TIMEV in RTC_SR depends on the value of TIMEVSEL.

Value	Name	Description
0	MINUTE	Minute change
1	HOUR	Hour change
2	MIDNIGHT	Every day at midnight
3	NOON	Every day at noon

- **CALEVSEL: Calendar Event Selection**

The event that generates the flag CALEV in RTC_SR depends on the value of CALEVSEL

Value	Name	Description
0	WEEK	Week change (every Monday at time 00:00:00)
1	MONTH	Month change (every 01 of each month at time 00:00:00)
2	YEAR	Year change (every January 1 at time 00:00:00)

Downloaded from Arrow.com.

26.6.2 RTC Mode Register

Name: RTC_MR

Address: 0x400E1864

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	TPERIOD		–	THIGH		
23	22	21	20	19	18	17	16
–	OUT1			–	OUT0		
15	14	13	12	11	10	9	8
HIGHPPM	CORRECTION						
7	6	5	4	3	2	1	0
–	–	–	NEGPPM	–	–	PERSIAN	HRMOD

This register can only be written if the WPEN bit is cleared in the RTC Write Protection Mode Register.

- **HRMOD: 12-/24-hour Mode**

0: 24-hour mode is selected.

1: 12-hour mode is selected.

- **PERSIAN: PERSIAN Calendar**

0: Gregorian calendar.

1: Persian calendar.

- **NEGPPM: NEGative PPM Correction**

0: Positive correction (the divider will be slightly higher than 32768).

1: Negative correction (the divider will be slightly lower than 32768).

Refer to CORRECTION and HIGHPPM field descriptions.

Note: NEGPPM must be cleared to correct a crystal slower than 32.768 kHz.

- **CORRECTION: Slow Clock Correction**

0: No correction

1–127: The slow clock will be corrected according to the formula given in HIGHPPM description.

- **HIGHPPM: HIGH PPM Correction**

0: Lower range ppm correction with accurate correction.

1: Higher range ppm correction with accurate correction.

If the absolute value of the correction to be applied is lower than 30 ppm, it is recommended to clear HIGHPPM. HIGHPPM set to 1 is recommended for 30 ppm correction and above.

Formula:

If HIGHPPM = 0, then the clock frequency correction range is from 1.5 ppm up to 98 ppm. The RTC accuracy is less than 1 ppm for a range correction from 1.5 ppm up to 30 ppm.

The correction field must be programmed according to the required correction in ppm; the formula is as follows:

$$CORRECTION = \frac{3906}{20 \times ppm} - 1$$

Downloaded from Arrow.com.

The value obtained must be rounded to the nearest integer prior to being programmed into CORRECTION field.

If HIGHPPM = 1, then the clock frequency correction range is from 30.5 ppm up to 1950 ppm. The RTC accuracy is less than 1 ppm for a range correction from 30.5 ppm up to 90 ppm.

The correction field must be programmed according to the required correction in ppm; the formula is as follows:

$$CORRECTION = \frac{3906}{ppm} - 1$$

The value obtained must be rounded to the nearest integer prior to be programmed into CORRECTION field.

If NEGPPM is set to 1, the ppm correction is negative (used to correct crystals that are faster than the nominal 32.768 kHz).

- **OUT0: RTCOUT0 OutputSource Selection**

Value	Name	Description
0	NO_WAVE	No waveform, stuck at ‘0’
1	FREQ1HZ	1 Hz square wave
2	FREQ32HZ	32 Hz square wave
3	FREQ64HZ	64 Hz square wave
4	FREQ512HZ	512 Hz square wave
5	ALARM_TOGGLE	Output toggles when alarm flag rises
6	ALARM_FLAG	Output is a copy of the alarm flag
7	PROG_PULSE	Duty cycle programmable pulse

- **OUT1: RTCOUT1 Output Source Selection**

Value	Name	Description
0	NO_WAVE	No waveform, stuck at ‘0’
1	FREQ1HZ	1 Hz square wave
2	FREQ32HZ	32 Hz square wave
3	FREQ64HZ	64 Hz square wave
4	FREQ512HZ	512 Hz square wave
5	ALARM_TOGGLE	Output toggles when alarm flag rises
6	ALARM_FLAG	Output is a copy of the alarm flag
7	PROG_PULSE	Duty cycle programmable pulse

- **THIGH: High Duration of the Output Pulse**

Value	Name	Description
0	H_31MS	31.2 ms
1	H_16MS	15.6 ms
2	H_4MS	3.91 ms
3	H_976US	976 µs
4	H_488US	488 µs

Downloaded from Arrow.com.

Value	Name	Description
5	H_122US	122 µs
6	H_30US	30.5 µs
7	H_15US	15.2 µs

- **TPERIOD: Period of the Output Pulse**

Value	Name	Description
0	P_1S	1 second
1	P_500MS	500 ms
2	P_250MS	250 ms
3	P_125MS	125 ms

Downloaded from Arrow.com.

26.6.3 RTC Time Register

Name: RTC_TIMR

Address: 0x400E1868

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	AMPM	HOUR					

15	14	13	12	11	10	9	8
–	MIN						

7	6	5	4	3	2	1	0
–	SEC						

- **SEC: Current Second**

The range that can be set is 0–59 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **MIN: Current Minute**

The range that can be set is 0–59 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **HOUR: Current Hour**

The range that can be set is 1–12 (BCD) in 12-hour mode or 0–23 (BCD) in 24-hour mode.

- **AMPM: Ante Meridiem Post Meridiem Indicator**

This bit is the AM/PM indicator in 12-hour mode.

0: AM.

1: PM.

Downloaded from Arrow.com.

26.6.4 RTC Calendar Register

Name: RTC_CALR

Address: 0x400E186C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	DATE					

23	22	21	20	19	18	17	16
DAY			MONTH				

15	14	13	12	11	10	9	8
YEAR							

7	6	5	4	3	2	1	0
–	CENT						

- **CENT: Current Century**

The range that can be set is 19–20 (Gregorian) or 13–14 (Persian) (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **YEAR: Current Year**

The range that can be set is 00–99 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **MONTH: Current Month**

The range that can be set is 01–12 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **DAY: Current Day in Current Week**

The range that can be set is 1–7 (BCD).

The coding of the number (which number represents which day) is user-defined as it has no effect on the date counter.

- **DATE: Current Day in Current Month**

The range that can be set is 01–31 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

Downloaded from Arrow.com.

26.6.5 RTC Time Alarm Register

Name: RTC_TIMALR

Address: 0x400E1870

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
HOUREN	AMPM	HOUR					
15	14	13	12	11	10	9	8
MINEN	MIN						
7	6	5	4	3	2	1	0
SECEN	SEC						

This register can only be written if the WPEN bit is cleared in the RTC Write Protection Mode Register.

Note: To change one of the SEC, MIN, HOUR fields, it is recommended to disable the field before changing the value and then re-enable it after the change has been made. This requires up to three accesses to the RTC_TIMALR. The first access clears the enable corresponding to the field to change (SECEN, MINEN, HOUREN). If the field is already cleared, this access is not required. The second access performs the change of the value (SEC, MIN, HOUR). The third access is required to re-enable the field by writing 1 in SECEN, MINEN, HOUREN fields.

- **SEC: Second Alarm**

This field is the alarm field corresponding to the BCD-coded second counter.

- **SECEN: Second Alarm Enable**

0: The second-matching alarm is disabled.

1: The second-matching alarm is enabled.

- **MIN: Minute Alarm**

This field is the alarm field corresponding to the BCD-coded minute counter.

- **MINEN: Minute Alarm Enable**

0: The minute-matching alarm is disabled.

1: The minute-matching alarm is enabled.

- **HOUR: Hour Alarm**

This field is the alarm field corresponding to the BCD-coded hour counter.

- **AMPM: AM/PM Indicator**

This field is the alarm field corresponding to the BCD-coded hour counter.

- **HOUREN: Hour Alarm Enable**

0: The hour-matching alarm is disabled.

1: The hour-matching alarm is enabled.

Downloaded from Arrow.com.

26.6.6 RTC Calendar Alarm Register

Name: RTC_CALALR

Address: 0x400E1874

Access: Read/Write

31	30	29	28	27	26	25	24
DATEEN	–	DATE					
23	22	21	20	19	18	17	16
MTHEN	–	–	MONTH				
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

This register can only be written if the WPEN bit is cleared in the RTC Write Protection Mode Register.

Note: To change one of the DATE, MONTH fields, it is recommended to disable the field before changing the value and then re-enable it after the change has been made. This requires up to three accesses to the RTC_CALALR. The first access clears the enable corresponding to the field to change (DATEEN, MTHEN). If the field is already cleared, this access is not required. The second access performs the change of the value (DATE, MONTH). The third access is required to re-enable the field by writing 1 in DATEEN, MTHEN fields.

- **MONTH: Month Alarm**

This field is the alarm field corresponding to the BCD-coded month counter.

- **MTHEN: Month Alarm Enable**

0: The month-matching alarm is disabled.

1: The month-matching alarm is enabled.

- **DATE: Date Alarm**

This field is the alarm field corresponding to the BCD-coded date counter.

- **DATEEN: Date Alarm Enable**

0: The date-matching alarm is disabled.

1: The date-matching alarm is enabled.

Downloaded from Arrow.com.

26.6.7 RTC Status Register

Name: RTC_SR

Address: 0x400E1878

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	TDERR	CALEV	TIMEV	SEC	ALARM	ACKUPD

- **ACKUPD: Acknowledge for Update**

Value	Name	Description
0	FREERUN	Time and calendar registers cannot be updated.
1	UPDATE	Time and calendar registers can be updated.

- **ALARM: Alarm Flag**

Value	Name	Description
0	NO_ALARMEVENT	No alarm matching condition occurred.
1	ALARMEVENT	An alarm matching condition has occurred.

- **SEC: Second Event**

Value	Name	Description
0	NO_SECEVENT	No second event has occurred since the last clear.
1	SECEVENT	At least one second event has occurred since the last clear.

- **TIMEV: Time Event**

Value	Name	Description
0	NO_TIMEVENT	No time event has occurred since the last clear.
1	TIMEVENT	At least one time event has occurred since the last clear.

Note: The time event is selected in the TIMEVSEL field in the Control Register (RTC_CR) and can be any one of the following events: minute change, hour change, noon, midnight (day change).

- **CALEV: Calendar Event**

Value	Name	Description
0	NO_CALEVENT	No calendar event has occurred since the last clear.
1	CALEVENT	At least one calendar event has occurred since the last clear.

Note: The calendar event is selected in the CALEVSEL field in the Control Register (RTC_CR) and can be any one of the following events: week change, month change and year change.

Downloaded from Arrow.com.

- **TDERR: Time and/or Date Free Running Error**

Value	Name	Description
0	CORRECT	The internal free running counters are carrying valid values since the last read of the Status Register (RTC_SR).
1	ERR_TIMEDATE	The internal free running counters have been corrupted (invalid date or time, non-BCD values) since the last read and/or they are still invalid.

Downloaded from Arrow.com.

26.6.8 RTC Status Clear Command Register

Name: RTC_SCCR

Address: 0x400E187C

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	TDERRCLR	CALCLR	TIMCLR	SECCLR	ALRCLR	ACKCLR

- **ACKCLR: Acknowledge Clear**

0: No effect.

1: Clears corresponding status flag in the Status Register (RTC_SR).

- **ALRCLR: Alarm Clear**

0: No effect.

1: Clears corresponding status flag in the Status Register (RTC_SR).

- **SECCLR: Second Clear**

0: No effect.

1: Clears corresponding status flag in the Status Register (RTC_SR).

- **TIMCLR: Time Clear**

0: No effect.

1: Clears corresponding status flag in the Status Register (RTC_SR).

- **CALCLR: Calendar Clear**

0: No effect.

1: Clears corresponding status flag in the Status Register (RTC_SR).

- **TDERRCLR: Time and/or Date Free Running Error Clear**

0: No effect.

1: Clears corresponding status flag in the Status Register (RTC_SR).

Downloaded from Arrow.com.

26.6.9 RTC Interrupt Enable Register

Name: RTC_IER

Address: 0x400E1880

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	TDERREN	CALEN	TIMEN	SECEN	ALREN	ACKEN

- **ACKEN: Acknowledge Update Interrupt Enable**

0: No effect.

1: The acknowledge for update interrupt is enabled.

- **ALREN: Alarm Interrupt Enable**

0: No effect.

1: The alarm interrupt is enabled.

- **SECEN: Second Event Interrupt Enable**

0: No effect.

1: The second periodic interrupt is enabled.

- **TIMEN: Time Event Interrupt Enable**

0: No effect.

1: The selected time event interrupt is enabled.

- **CALEN: Calendar Event Interrupt Enable**

0: No effect.

1: The selected calendar event interrupt is enabled.

- **TDERREN: Time and/or Date Error Interrupt Enable**

0: No effect.

1: The time and date error interrupt is enabled.

Downloaded from Arrow.com.

26.6.10 RTC Interrupt Disable Register

Name: RTC_IDR

Address: 0x400E1884

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	TDERRDIS	CALDIS	TIMDIS	SECDIS	ALRDIS	ACKDIS

- **ACKDIS: Acknowledge Update Interrupt Disable**

0: No effect.

1: The acknowledge for update interrupt is disabled.

- **ALRDIS: Alarm Interrupt Disable**

0: No effect.

1: The alarm interrupt is disabled.

- **SECDIS: Second Event Interrupt Disable**

0: No effect.

1: The second periodic interrupt is disabled.

- **TIMDIS: Time Event Interrupt Disable**

0: No effect.

1: The selected time event interrupt is disabled.

- **CALDIS: Calendar Event Interrupt Disable**

0: No effect.

1: The selected calendar event interrupt is disabled.

- **TDERRDIS: Time and/or Date Error Interrupt Disable**

0: No effect.

1: The time and date error interrupt is disabled.

Downloaded from Arrow.com.

26.6.11 RTC Interrupt Mask Register

Name: RTC_IMR

Address: 0x400E1888

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	TDERR	CAL	TIM	SEC	ALR	ACK

- **ACK: Acknowledge Update Interrupt Mask**

0: The acknowledge for update interrupt is disabled.

1: The acknowledge for update interrupt is enabled.

- **ALR: Alarm Interrupt Mask**

0: The alarm interrupt is disabled.

1: The alarm interrupt is enabled.

- **SEC: Second Event Interrupt Mask**

0: The second periodic interrupt is disabled.

1: The second periodic interrupt is enabled.

- **TIM: Time Event Interrupt Mask**

0: The selected time event interrupt is disabled.

1: The selected time event interrupt is enabled.

- **CAL: Calendar Event Interrupt Mask**

0: The selected calendar event interrupt is disabled.

1: The selected calendar event interrupt is enabled.

- **TDERR: Time and/or Date Error Mask**

0: The time and/or date error event is disabled.

1: The time and/or date error event is enabled.

Downloaded from Arrow.com.

26.6.12 RTC Valid Entry Register

Name: RTC_VER

Address: 0x400E188C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	NVCALALR	NVTIMALR	NVCAL	NVTIM

- **NVTIM: Non-valid Time**

0: No invalid data has been detected in RTC_TIMR (Time Register).

1: RTC_TIMR has contained invalid data since it was last programmed.

- **NVCAL: Non-valid Calendar**

0: No invalid data has been detected in RTC_CALR (Calendar Register).

1: RTC_CALR has contained invalid data since it was last programmed.

- **NVTIMALR: Non-valid Time Alarm**

0: No invalid data has been detected in RTC_TIMALR (Time Alarm Register).

1: RTC_TIMALR has contained invalid data since it was last programmed.

- **NVCALALR: Non-valid Calendar Alarm**

0: No invalid data has been detected in RTC_CALALR (Calendar Alarm Register).

1: RTC_CALALR has contained invalid data since it was last programmed.

Downloaded from Arrow.com.

26.6.13 RTC Write Protection Mode Register

Name: RTC_WPMR

Address: 0x400E1944

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							

23	22	21	20	19	18	17	16
WPKEY							

15	14	13	12	11	10	9	8
WPKEY							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x525443 (“RTC” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x525443 (“RTC” in ASCII).

The following registers can be write-protected:

- RTC Mode Register
- RTC Time Alarm Register
- RTC Calendar Alarm Register

- **WPKEY: Write Protection Key**

Value	Name	Description
0x525443	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

Downloaded from Arrow.com.

27. Real-time Timer (RTT)

27.1 Description

The Real-timeTimer (RTT) is built around a 32-bit counter used to count roll-over events of the programmable 16-bit prescaler driven from the 32-kHz slow clock source. It generates a periodic interrupt and/or triggers an alarm on a programmed value.

The RTT can also be configured to be driven by the 1Hz RTC signal, thus taking advantage of a calibrated 1Hz clock.

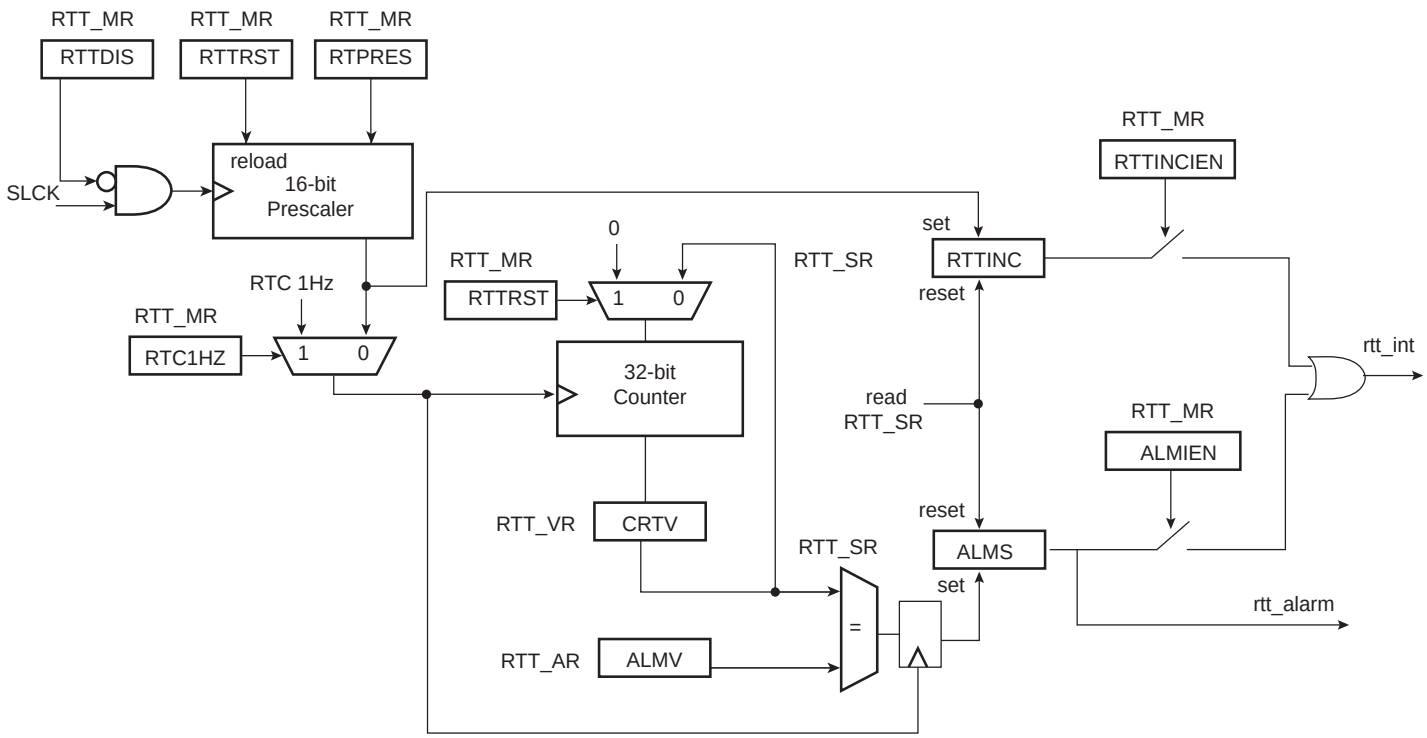
The slow clock source can be fully disabled to reduce power consumption when only an elapsed seconds count is required.

27.2 Embedded Characteristics

- 32-bit Free-running Counter on prescaled slow clock or RTC calibrated 1Hz clock
- 16-bit Configurable Prescaler
- Interrupt on Alarm or Counter Increment

27.3 Block Diagram

Figure 27-1. Real-time Timer



Downloaded from Arrow.com.

27.4 Functional Description

The programmable 16-bit prescaler value can be configured through the RTPRES field in the "Real-timeTimer Mode Register" (RTT_MR).

Configuring the RTPRES field value to 0x8000 (default value) corresponds to feeding the real-time counter with a 1Hz signal (if the slow clock is 32.768 kHz). The 32-bit counter can count up to 2^{32} seconds, corresponding to more than 136 years, then roll over to 0. Bit RTTINC in the "Real-time Timer Status Register" (RTT_SR) is set each time there is a prescaler roll-over (see Figure 27-2)

The real-time 32-bit counter can also be supplied by the 1Hz RTC clock. This mode is interesting when the RTC 1Hz is calibrated (CORRECTION field ≠ 0 in RTC_MR) in order to guaranty the synchronism between RTC and RTT counters.

Setting the RTC1HZ bit in the RTT_MR drives the 32-bit RTT counter from the 1Hz RTC clock. In this mode, the RTPRES field has no effect on the 32-bit counter.

The prescaler roll-over generates an increment of the real-time timer counter if RTC1HZ = 0. Otherwise, if RTC1HZ = 1, the real-time timer counter is incremented every second. The RTTINC bit is set independently from the 32-bit counter increment.

The real-time timer can also be used as a free-running timer with a lower time-base. The best accuracy is achieved by writing RTPRES to 3 in RTT_MR.

Programming RTPRES to 1 or 2 is forbidden.

If the RTT is configured to trigger an interrupt, the interrupt occurs two slow clock cycles after reading the RTT_SR. To prevent several executions of the interrupt handler, the interrupt must be disabled in the interrupt handler and re-enabled when the RTT_SR is cleared.

The CRTV field can be read at any time in the "Real-time Timer Value Register" (RTT_VR). As this value can be updated asynchronously with the Master Clock, the CRTV field must be read twice at the same value to read a correct value.

The current value of the counter is compared with the value written in the "Real-time Timer Alarm Register" (RTT_AR). If the counter value matches the alarm, the ALMS bit in the RTT_SR is set. The RTT_AR is set to its maximum value (0xFFFF_FFFF) after a reset.

The ALMS flag is always a source of the RTT alarm signal that may be used to exit the system from low power modes (see Figure 27-1).

The alarm interrupt must be disabled (ALMIEN must be cleared in RTT_MR) when writing a new ALMV value in the RTT_AR.

The RTTINC bit can be used to start a periodic interrupt, the period being one second when the RTPRES field value = 0x8000 and the slow clock = 32.768 kHz.

The RTTINCIEN bit must be cleared prior to writing a new RTPRES value in the RTT_MR.

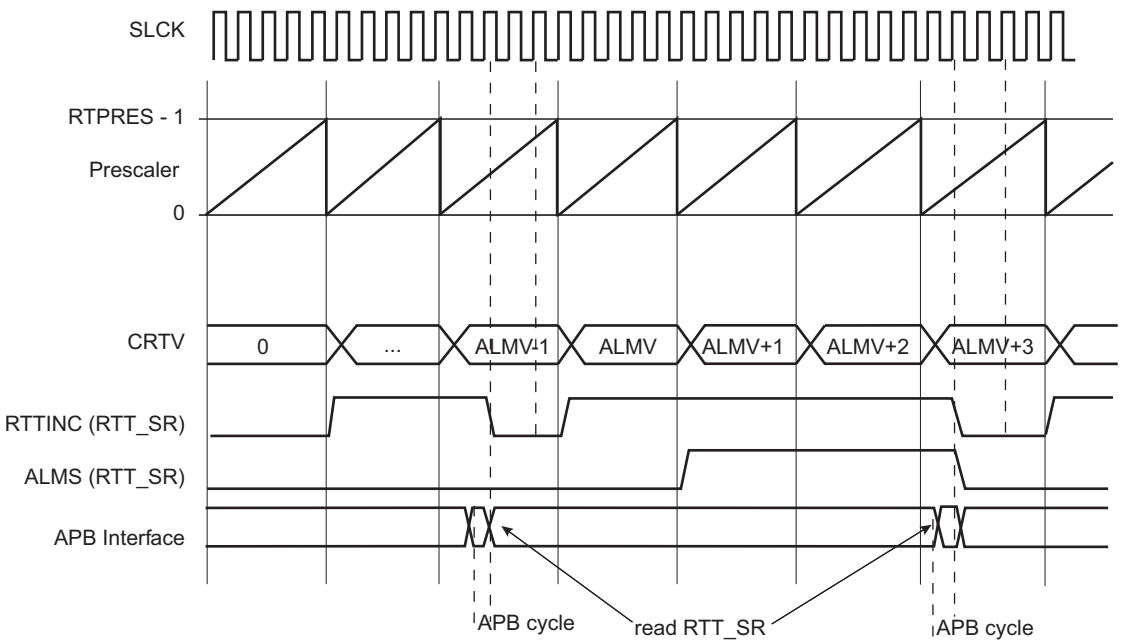
Reading the RTT_SR automatically clears the RTTINC and ALMS bits.

Writing the RTTRST bit in the RTT_MR immediately reloads and restarts the clock divider with the new programmed value. This also resets the 32-bit counter.

When not used, the Real-time Timer can be disabled in order to suppress dynamic power consumption in this module. This can be achieved by setting the RTTDIS bit in the RTT_MR.

Downloaded from Arrow.com.

Figure 27-2. RTT Counting



Downloaded from Arrow.com.

27.5 Real-time Timer (RTT) User Interface

Table 27-1. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Mode Register	RTT_MR	Read/Write	0x0000_8000
0x04	Alarm Register	RTT_AR	Read/Write	0xFFFF_FFFF
0x08	Value Register	RTT_VR	Read-only	0x0000_0000
0x0C	Status Register	RTT_SR	Read-only	0x0000_0000

Downloaded from Arrow.com.

27.5.1 Real-timeTimer Mode Register

Name: RTT_MR

Address: 0x400E1830

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	RTC1HZ

23	22	21	20	19	18	17	16
–	–	–	RTTDIS	–	RTTRST	RTTINCIEN	ALMIEN

15	14	13	12	11	10	9	8
RTPRES							

7	6	5	4	3	2	1	0
RTPRES							

- **RTPRES: Real-time Timer Prescaler Value**

Defines the number of SLCK periods required to increment the real-time timer. RTPRES is defined as follows:

RTPRES = 0: The prescaler period is equal to 2^{16} * SLCK periods.

RTPRES = 1 or 2: forbidden.

RTPRES ≠ 0,1 or 2: The prescaler period is equal to RTPRES * SLCK periods.

Note: The RTTINCIEN bit must be cleared prior to writing a new RTPRES value.

- **ALMIEN: Alarm Interrupt Enable**

0: The bit ALMS in RTT_SR has no effect on interrupt.

1: The bit ALMS in RTT_SR asserts interrupt.

- **RTTINCIEN: Real-time Timer Increment Interrupt Enable**

0: The bit RTTINC in RTT_SR has no effect on interrupt.

1: The bit RTTINC in RTT_SR asserts interrupt.

- **RTTRST: Real-time Timer Restart**

0: No effect.

1: Reloads and restarts the clock divider with the new programmed value. This also resets the 32-bit counter.

- **RTTDIS: Real-time Timer Disable**

0: The real-time timer is enabled.

1: The real-time timer is disabled (no dynamic power consumption).

- **RTC1HZ: Real-time Clock 1Hz Clock Selection**

0: The RTT 32-bit counter is driven by the 16-bit prescaler roll-over events.

1: The RTT 32-bit counter is driven by the 1Hz RTC clock.

Downloaded from Arrow.com.

27.5.2 Real-time Timer Alarm Register

Name: RTT_AR

Address: 0x400E1834

Access: Read/Write

31	30	29	28	27	26	25	24
ALMV							

23	22	21	20	19	18	17	16
ALMV							

15	14	13	12	11	10	9	8
ALMV							

7	6	5	4	3	2	1	0
ALMV							

- **ALMV: Alarm Value**

When the CRTV value in RTT_VR equals the ALMV field, the ALMS flag is set in RTT_SR. As soon as the ALMS flag rises, the CRTV value equals ALMV+1 (refer to Figure 27-2).

Note: The alarm interrupt must be disabled (ALMIEN must be cleared in RTT_MR) when writing a new ALMV value.

Downloaded from Arrow.com.

27.5.3 Real-time Timer Value Register

Name: RTT_VR

Address: 0x400E1838

Access: Read-only

31	30	29	28	27	26	25	24
CRTV							

23	22	21	20	19	18	17	16
CRTV							

15	14	13	12	11	10	9	8
CRTV							

7	6	5	4	3	2	1	0
CRTV							

- **CRTV: Current Real-time Value**

Returns the current value of the Real-time Timer.

Note: As CRTV can be updated asynchronously, it must be read twice at the same value.

Downloaded from Arrow.com.

27.5.4 Real-time Timer Status Register

Name: RTT_SR

Address: 0x400E183C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	RTTINC	ALMS

- **ALMS: Real-time Alarm Status (cleared on read)**

0: The real-time alarm has not occurred since the last read of RTT_SR.

1: The real-time alarm occurred since the last read of RTT_SR.

- **RTTINC: Prescaler Roll-over Status (cleared on read)**

0: No prescaler roll-over occurred since the last read of the RTT_SR.

1: Prescaler roll-over occurred since the last read of the RTT_SR.

Downloaded from Arrow.com.

28. SDRAM Controller (SDRAMC)

28.1 Description

The SDRAM Controller (SDRAMC) extends the memory capabilities of a chip by providing the interface to external 16-bitDRAM devices. The page size supports ranges from 2048 to 8192 and the number of columns from 256 to 2048. It supports byte (8-bit), half-word (16-bit) and word (32-bit) accesses.

The SDRAMC supports a read or write burst length of one location. It keeps track of the active row in each bank, thus maximizing SDRAM performance, e.g., the application may be placed in one bank and data in the other banks. For optimized performance, it is advisable to avoid accessing different rows in the same bank.

The SDRAMC supports a CAS latency of 1, 2 or 3 and optimizes the read access depending on the frequency.

The different modes available – Self-refresh, Power-down and Deep Power-down modes – minimize power consumption on the SDRAM device.

28.2 Embedded Characteristics

- Numerous Configurations Supported
  - 2K, 4K, 8K Row Address Memory Parts
  - SDRAM with Two or Four Internal Banks
  - SDRAM with 16-bit Data Path
- Programming Facilities
  - Word, Half-word, Byte Access
  - Automatic Page Break When Memory Boundary Has Been Reached
  - Multibank Ping-pong Access
  - Timing Parameters Specified by Software
  - Automatic Refresh Operation, Refresh Rate is Programmable
  - Automatic Update of DS, TCR and PASR Parameters (Mobile SDRAM Devices)
- Energy-saving Capabilities
  - Self-refresh, Power-down and Deep Power Modes Supported
  - Supports Mobile SDRAM Devices
- Error Detection
  - Refresh Error Interrupt
- SDRAM Power-up Initialization by Software
- CAS Latency of 1, 2, 3 Supported
- Auto Precharge Command Not Used
- Zero Wait State Scrambling/Unscrambling Function with User Key

Downloaded from Arrow.com.

28.3 Signal Description

Table 28-1. Signal Description

Name	Description	Type	Active Level
SDCK	SDRAM Clock	Output	–
SDCKE	SDRAM Clock Enable	Output	High
SDCS	SDRAMC Chip Select	Output	Low
BA[1:0]	Bank Select Signals	Output	–
RAS	Row Signal	Output	Low
CAS	Column Signal	Output	Low
SDWE	SDRAM Write Enable	Output	Low
NBS[1:0]	Data Mask Enable Signals	Output	Low
SDRAMC_A[12:0]	Address Bus	Output	–
D[15:0]	Data Bus	I/O	–

Downloaded from Arrow.com.

28.4 Software Interface/SDRAM Organization, Address Mapping

The SDRAM address space is organized into banks, rows, and columns. The SDRAMC allows mapping different memory types according to the values set in the SDRAMC Configuration Register (SDRAMC_CR).

The SDRAMC makes the SDRAM device access protocol transparent to the user. Table 28-2 to Table 28-4 illustrate the SDRAM device memory mapping seen by the user in correlation with the device structure. Various configurations are illustrated.

28.4.1 SDRAM Address Mapping for 16-bit Memory Data Bus Width

Table 28-2. SDRAM Configuration Mapping: 2K Rows, 256/512/1024/2048 Columns

CPU Address Line																										
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
						Bk[1:0]	Row[10:0]												Column[7:0]								M0
					Bk[1:0]	Row[10:0]												Column[8:0]									M0
				Bk[1:0]	Row[10:0]												Column[9:0]										M0
			Bk[1:0]	Row[10:0]												Column[10:0]											M0

Table 28-3. SDRAM Configuration Mapping: 4K Rows, 256/512/1024/2048 Columns

CPU Address Line																										
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
					Bk[1:0]	Row[11:0]												Column[7:0]									M0
				Bk[1:0]	Row[11:0]												Column[8:0]										M0
			Bk[1:0]	Row[11:0]												Column[9:0]											M0
		Bk[1:0]	Row[11:0]													Column[10:0]											M0

Table 28-4. SDRAM Configuration Mapping: 8K Rows, 256/512/1024/2048 Columns

CPU Address Line																										
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
				Bk[1:0]	Row[12:0]												Column[7:0]										M0
			Bk[1:0]	Row[12:0]												Column[8:0]											M0
		Bk[1:0]	Row[12:0]													Column[9:0]											M0
	Bk[1:0]	Row[12:0]														Column[10:0]											M0

Notes: 1. M0 is the byte address inside a 16-bit half-word.
2. Bk[1] = BA1, Bk[0] = BA0.

Downloaded from Arrow.com.

28.5 Product Dependencies

28.5.1 SDRAM Device Initialization

The initialization sequence is generated by software. The SDRAM devices are initialized by the following sequence:

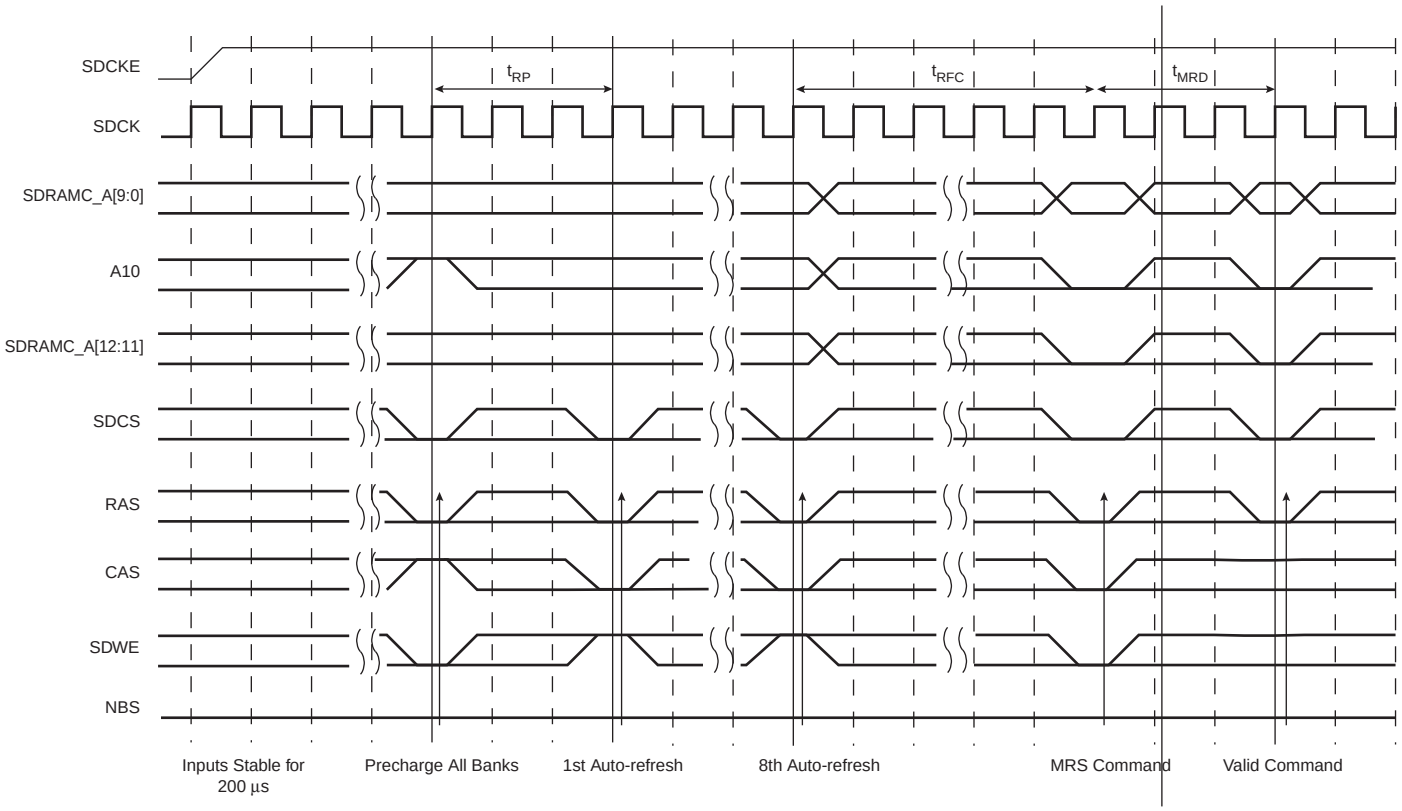
1. Set the SDRAM features in the SDRAMC_CR: asynchronous timings (TRC, TRAS, etc.), number of columns, number of rows, CAS latency and data bus width. For mobile SDRAM, configure temperature-compensated self-refresh (TCSR), drive strength (DS) and partial array self-refresh (PASR) in the Low Power Register (SDRAMC_LPR).
2. Select the SDRAM memory device type in the Memory Device Register (SDRAMC_MDR).
3. A pause of at least 200 µs must be observed before a signal toggle.
4. [(1)]A NOP command is issued to the SDRAM devices. The application must write a 1 to the MODE field in the Mode Register (SDRAMC_MR). Read the SDRAMC_MR and add a memory barrier assembler instruction just after the read. Perform a write access to any SDRAM address.
5. An All Banks Precharge command is issued to the SDRAM. The application must write a 2 to the MODE field in the SDRAMC_MR. Read the SDRAMC_MR and add a memory barrier assembler instruction just after the read. Perform a write access to any SDRAM address.
6. Eight auto-refresh (CBR) cycles are provided. The application must set the MODE field to 4 in the SDRAMC_MR. Read the SDRAMC_MR and add a memory barrier assembler instruction just after the read. Perform a write access to any SDRAM location eight times.
7. A Mode Register set (MRS) cycle is issued to program the parameters of the SDRAM, in particular CAS latency and burst length. The application must write a 3 to the MODE field in the SDRAMC_MR. Read the SDRAMC_MR and add a memory barrier assembler instruction just after the read. Perform a write access to the SDRAM. The write address must be chosen so that BA[1:0] are set to 0. For example, with a 16-bit 128 MB SDRAM (12 rows, 9 columns, 4 banks) bank address, the SDRAM write access should be done at the address 0x70000000.
8. For mobile SDRAM initialization, an Extended Mode Register set (EMRS) cycle is issued to program the SDRAM parameters (TCSR, PASR, DS). The application must set the MODE field to 5 in the SDRAMC_MR. Read the SDRAMC_MR and add a memory barrier assembler instruction just after the read. Perform a write access to the SDRAM. The write address must be chosen so that BA[1] or BA[0] are set to 1. For example, with a 16-bit 128 MB SDRAM (12 rows, 9 columns, 4 banks) bank address, the SDRAM write access should be done at address 0x70800000 or 0x70400000.
9. The application must go into Normal mode. Configure MODE to 0 in the SDRAMC_MR. Read the SDRAMC_MR and add a memory barrier assembler instruction just after the read. Perform a write access at any location in the SDRAM.
10. Write the refresh rate into the COUNT field in the SDRAMC Refresh Timer Register (SDRAMC_TR). (Refresh rate = delay between refresh cycles). The SDRAM device requires a refresh every 15.625 µs or 7.81 µs. With a 100 MHz frequency, the Refresh Timer Register must be set with the value 1562 (15.625 µs x 100 MHz) or 781 (7.81 µs x 100 MHz).

After initialization, the SDRAM devices are fully functional.

Note: 1. The instructions stated in Step 4 of the initialization process must be respected to make sure the subsequent commands issued by the SDRAMC are taken into account.

Downloaded from Arrow.com.

Figure 28-1. SDRAM Device Initialization Sequence



28.5.2 I/O Lines

The pins used for interfacing the SDRAMC may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the SDRAMC pins to their peripheral function. If I/O lines of the SDRAMC are not used by the application, they can be used for other purposes by the PIO Controller.

Table 28-5. I/O Lines

Instance	Signal	I/O Line	Peripheral
SDRAMC	A0/NBS0	PC18	A
SDRAMC	A1	PC19	A
SDRAMC	A2	PC20	A
SDRAMC	A3	PC21	A
SDRAMC	A4	PC22	A
SDRAMC	A5	PC23	A
SDRAMC	A6	PC24	A
SDRAMC	A7	PC25	A
SDRAMC	A8	PC26	A
SDRAMC	A9	PC27	A
SDRAMC	A10	PC28	A
SDRAMC	A11	PC29	A
SDRAMC	A12	PC30	A

Downloaded from Arrow.com.

Table 28-5. I/O Lines

SDRAMC	A13	PC31	A
SDRAMC	A14	PA18	C
SDRAMC	A15	PA19	C
SDRAMC	A16/BA0	PA20	C
SDRAMC	A17/BA1	PA0	C
SDRAMC	A18	PA1	C
SDRAMC	A19	PA23	C
SDRAMC	A20	PA24	C
SDRAMC	A21/NANDALE	PC16	A
SDRAMC	A22/NANDCLE	PC17	A
SDRAMC	A23	PA25	C
SDRAMC	CAS	PD17	C
SDRAMC	D0	PC0	A
SDRAMC	D1	PC1	A
SDRAMC	D2	PC2	A
SDRAMC	D3	PC3	A
SDRAMC	D4	PC4	A
SDRAMC	D5	PC5	A
SDRAMC	D6	PC6	A
SDRAMC	D7	PC7	A
SDRAMC	D8	PE0	A
SDRAMC	D9	PE1	A
SDRAMC	D10	PE2	A
SDRAMC	D11	PE3	A
SDRAMC	D12	PE4	A
SDRAMC	D13	PE5	A
SDRAMC	D14	PA15	A
SDRAMC	D15	PA16	A
SDRAMC	NANDOE	PC9	A
SDRAMC	NANDWE	PC10	A
SDRAMC	NCS0	PC14	A
SDRAMC	NCS1/SDCS	PC15	A
SDRAMC	NCS1/SDCS	PD18	A
SDRAMC	NCS2	PA22	C
SDRAMC	NCS3	PC12	A
SDRAMC	NCS3	PD19	A
SDRAMC	NRD	PC11	A
SDRAMC	NWAIT	PC13	A

Downloaded from Arrow.com.

Table 28-5. I/O Lines

SDRAMC	NWR0/NWE	PC8	A
SDRAMC	NWR1/NBS1	PD15	C
SDRAMC	RAS	PD16	C
SDRAMC	SDA10	PC13	C
SDRAMC	SDA10	PD13	C
SDRAMC	SDCK	PD23	C
SDRAMC	SDCKE	PD14	C
SDRAMC	SDWE	PD29	C

28.5.3 Power Management

The SDRAMC may be clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the SDRAMC clock.

The SDRAM clock on pin SDCK is output as soon as the first access to the SDRAM is made during the initialization phase. To stop the SDRAM clock signal, the SDRAMC_LPR must be programmed with the self-refresh command.

28.5.4 Interrupt Sources

The SDRAMC interrupt (Refresh Error notification) is connected to the memory controller. This interrupt may be ORed with other system peripheral interrupt lines and is finally provided as the system interrupt source (Source 1) to the interrupt controller.

Using the SDRAMC interrupt requires the interrupt controller to be programmed first.

Table 28-6. Peripheral IDs

Instance	ID
SDRAMC	62

Downloaded from Arrow.com.

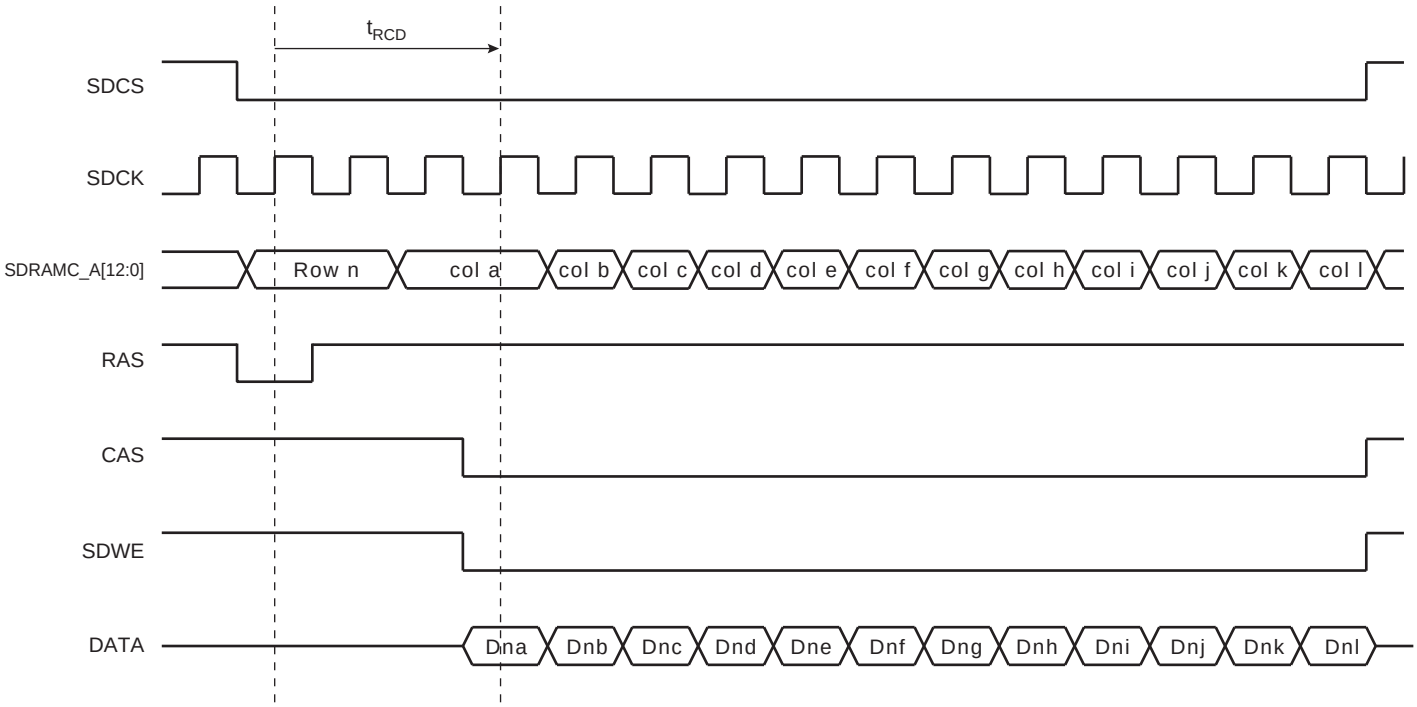
28.6 Functional Description

28.6.1 SDRAM Controller Write Cycle

The SDRAMC allows burst access or single access. In both cases, the SDRAMC keeps track of the active row in each bank, thus maximizing performance. To initiate a burst access, the SDRAMC uses the transfer type signal provided by the master requesting the access. If the next access is a sequential write access, writing to the SDRAM device is carried out. If the next access is a write-sequential access, but the current access is to a boundary page, or if the next access is in another row, then the SDRAMC generates a precharge command, activates the new row and initiates a write command. To comply with SDRAM timing parameters, additional clock cycles are inserted between precharge and active commands (t_{RP}), and between active and write commands (t_{RCD}) . For definition of these timing parameters, refer to the SDRAMC Configuration Register.

Refer to Figure 28-2.

Figure 28-2. Write Burst SDRAM Access



Downloaded from Arrow.com.

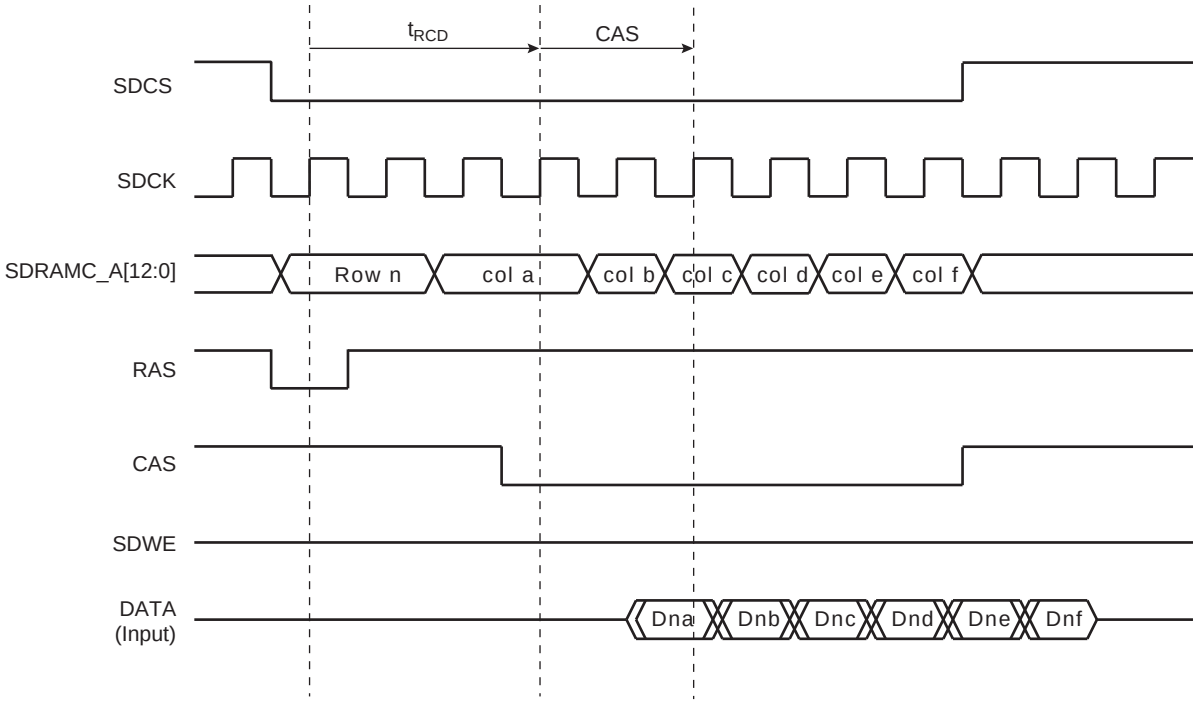
28.6.2 SDRAM Controller Read Cycle

The SDRAMC allows burst access, incremental burst of unspecified length or single access. In all cases, the SDRAMC keeps track of the active row in each bank, thus maximizing performance of the SDRAM. If row and bank addresses do not match the previous row/bank address, then the SDRAMC automatically generates a precharge command, activates the new row and starts the read command. To comply with the SDRAM timing parameters, additional clock cycles on SDCK are inserted between precharge and active commands (t_{RP}), and between active and read commands (t_{RCD}). These two parameters are set in the SDRAMC_CR. After a read command, additional wait states are generated to comply with the CAS latency (1, 2 or 3 clock delays specified in the SDRAMC_CR).

For a single access or an incremented burst of unspecified length, the SDRAMC anticipates the next access. While the last value of the column is returned by the SDRAMC on the bus, the SDRAMC anticipates the read to the next column and thus anticipates the CAS latency. This reduces the effect of the CAS latency on the internal bus.

For burst access of specified length (4, 8, 16 words), access is not anticipated. This case leads to the best performance. If the burst is broken (border, Busy mode, etc.), the next access is handled as an incrementing burst of unspecified length.

Figure 28-3. Read Burst SDRAM Access

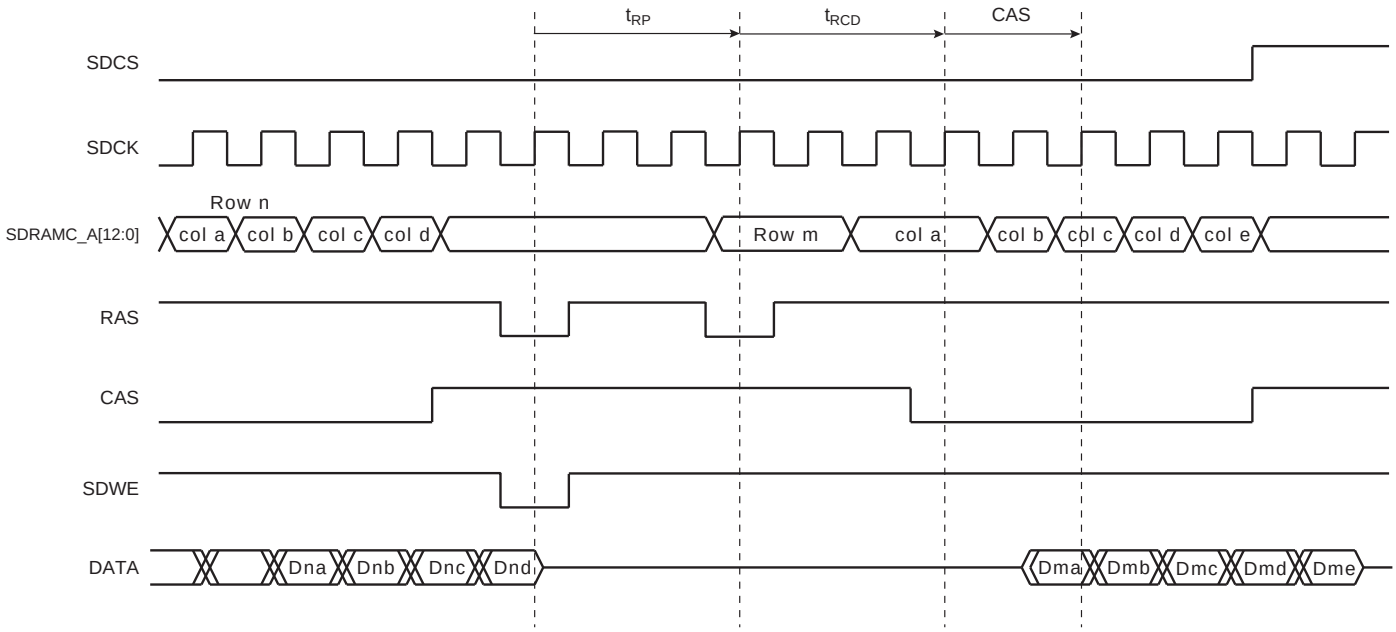


Downloaded from Arrow.com.

28.6.3 Border Management

When the memory row boundary has been reached, an automatic page break is inserted. In this case, the SDRAMC generates a precharge command, activates the new row and initiates a read or write command. To comply with SDRAM timing parameters, an additional clock cycle is inserted between the precharge and the active command (t_{RP}) and between the active and the read command (t_{RCD}). Refer to Figure 28-4.

Figure 28-4. Read Burst with Boundary Row Access



Downloaded from Arrow.com.

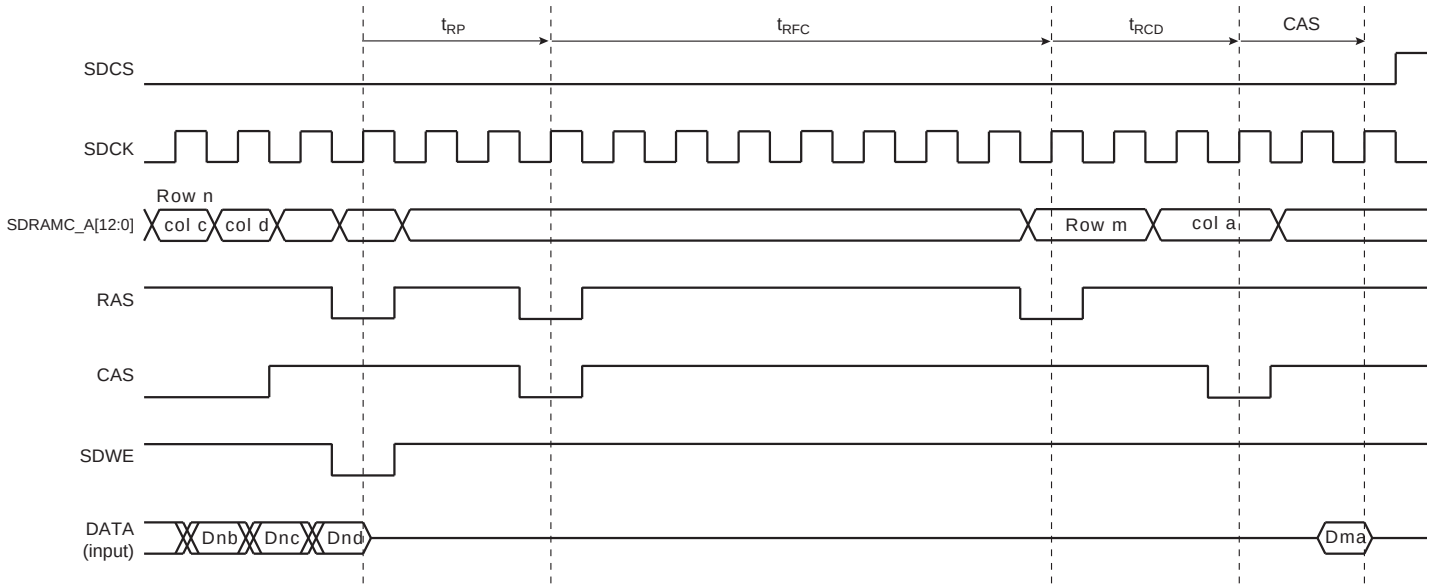
28.6.4 SDRAM Controller Refresh Cycles

An auto-refresh command is used to refresh the SDRAM device. Refresh addresses are generated internally by the SDRAM device and incremented after each auto-refresh automatically. The SDRAMC generates these auto-refresh commands periodically. An internal timer is loaded with the value in SDRAMC_TR that indicates the number of clock cycles between refresh cycles.

A refresh error interrupt is generated when the previous auto-refresh command did not perform. It is acknowledged by reading the Interrupt Status Register (SDRAMC_ISR).

When the SDRAMC initiates a refresh of the SDRAM device, internal memory accesses are not delayed. However, if the processor tries to access the SDRAM, the slave indicates that the device is busy and the master is held by a wait signal. Refer to Figure 28-5.

Figure 28-5. Refresh Cycle Followed by a Read Access



28.6.5 Power Management

Three low-power modes are available:

- Self-refresh mode: The SDRAM executes its own Auto-refresh cycle without control of the SDRAMC. Current drained by the SDRAM is very low.
- Power-down mode: Auto-refresh cycles are controlled by the SDRAMC. Between auto-refresh cycles, the SDRAM is in power-down. Current drained in Power-down mode is higher than in Self-refresh Mode.
- Deep Power-down mode (only available with Mobile SDRAM): The SDRAM contents are lost, but the SDRAM does not drain any current.

The SDRAMC activates one low-power mode as soon as the SDRAM device is not selected. It is possible to delay the entry in Self-refresh and Power-down modes after the last access by programming a timeout value in the SDRAMC_LPR.

28.6.5.1 Self-refresh Mode

This mode is selected by configuring the LPCB field to 1 in SDRAMC_LPR. In Self-refresh mode, the SDRAM device retains data without external clocking and provides its own internal clocking, thus performing its own auto-refresh cycles. All the inputs to the SDRAM device become “don’t care” except SDCKE, which remains low. As soon as the SDRAM device is selected, the SDRAMC provides a sequence of commands and exits Self-refresh mode.

Downloaded from Arrow.com.

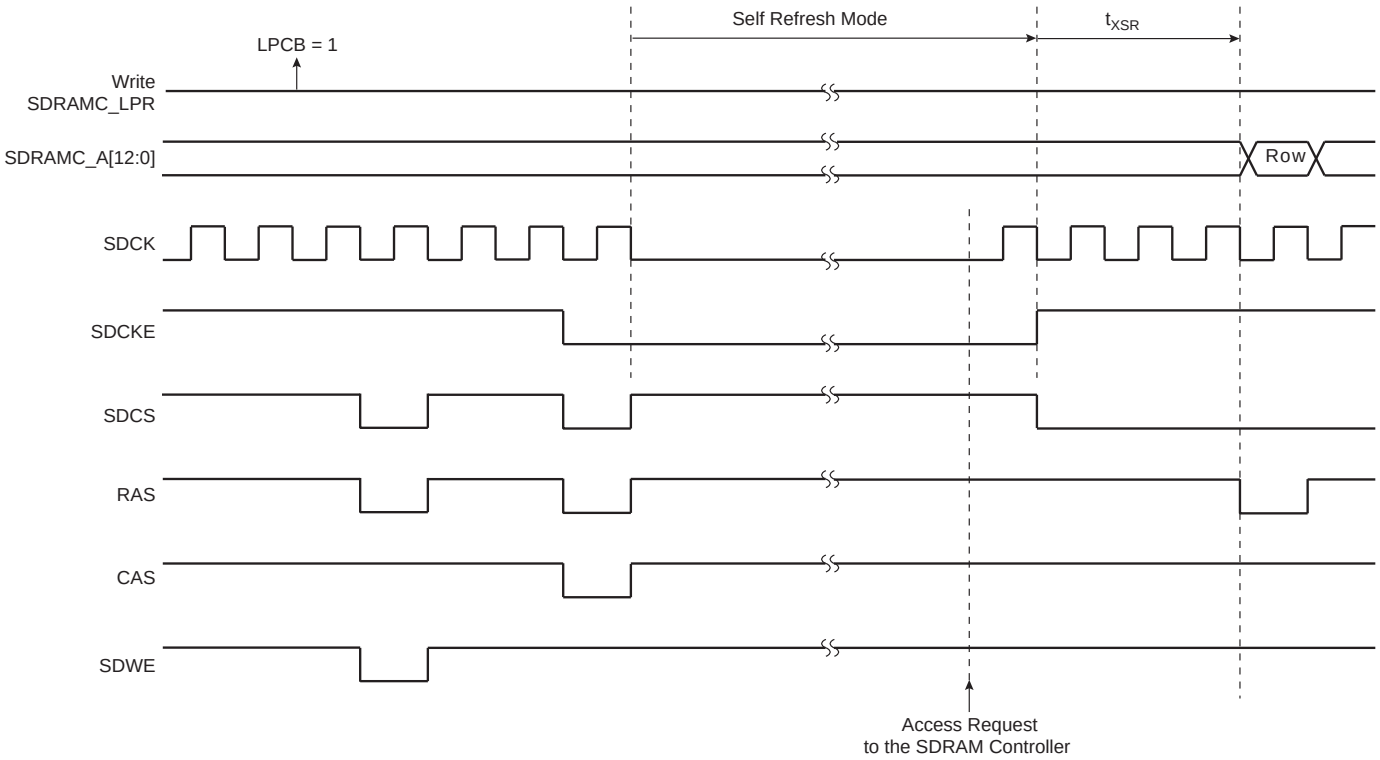
Some low-power SDRAMs (e.g., mobile SDRAM) can refresh only one-quarter or a half quarter or all banks of the SDRAM array. This feature reduces the self-refresh current. To configure this feature, Temperature Compensated Self-Refresh (TCSR), Partial Array Self-Refresh (PASR) and Drive Strength (DS) parameters must be set in the SDRAMC_LPR and transmitted to the low-power SDRAM during initialization.

After initialization, as soon as the PASR/DS/TCSR fields are modified and Self-refresh mode is activated, the Extended Mode Register is accessed automatically and the PASR/DS/TCSR bits are updated before entry into Self-refresh mode. This feature is not supported when SDRAMC shares an external bus with another controller.

The SDRAM device must remain in Self-refresh mode for a minimum period of t_{RAS} and may remain in Self-refresh mode for an indefinite period. Refer to Figure 28-6.

Note: Some SDRAM providers impose some cycles of burst auto-refresh immediately before self-refresh entry and immediately after self-refresh exit. For example, a SDRAM with 4096 rows will impose 4096 cycles of burst auto-refresh. This constraint is not supported.

Figure 28-6. Self-refresh Mode Behavior



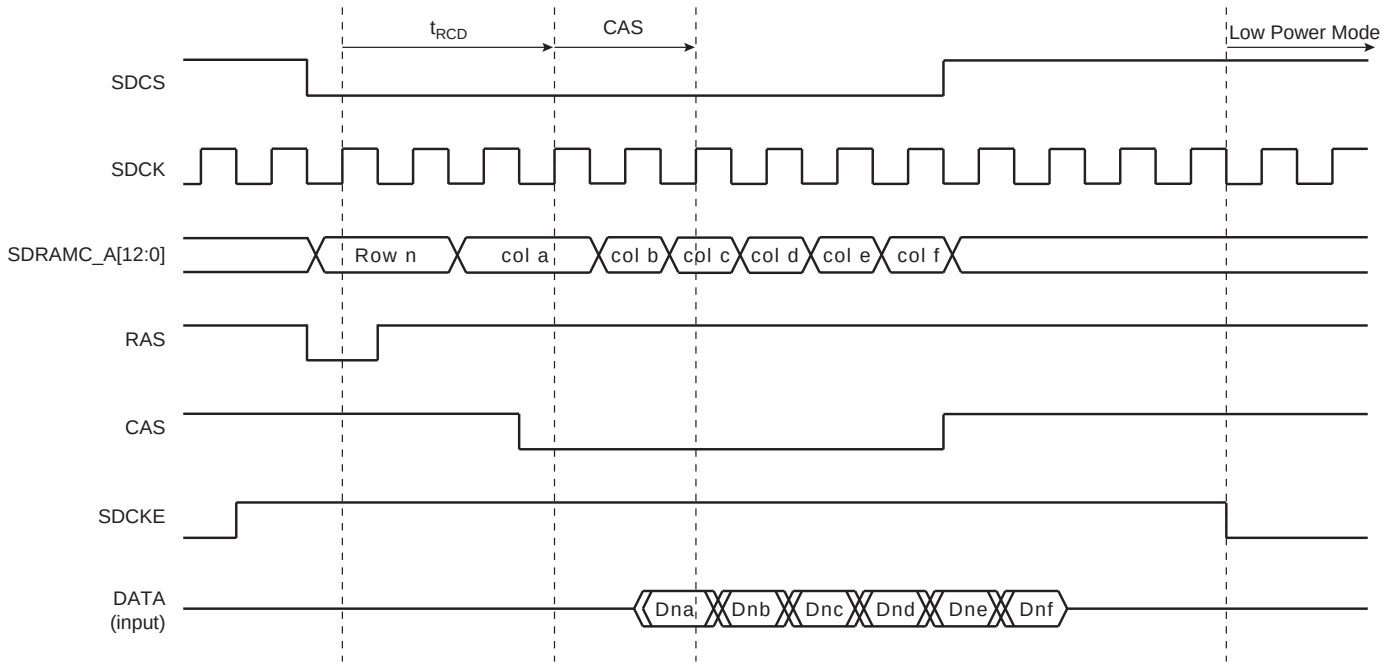
Downloaded from Arrow.com.

28.6.5.2 Low-power Mode

This mode is selected by configuring the LPCB field to 2 in the SDRAMC_LPR. Power consumption is greater than in Self-refresh mode. All the input and output buffers of the SDRAM device are deactivated except SDCKE, which remains low. In contrast to Self-refresh mode, the SDRAM device cannot remain in Low-power mode longer than the refresh period (64 ms for a whole device refresh operation). As no auto-refresh operations are performed by the SDRAM itself, the SDRAMC carries out the refresh operation. The exit procedure is faster than in Self-refresh mode.

Refer to Figure 28-7.

Figure 28-7. Low-power Mode Behavior



Downloaded from Arrow.com.

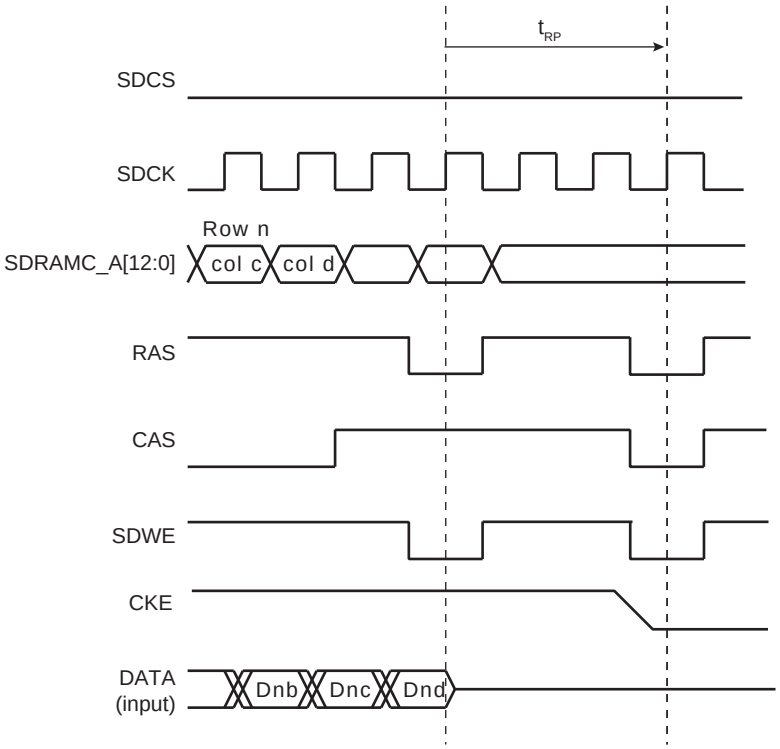
28.6.5.3 Deep Power-down Mode

This mode is selected by configuring the LPCB field to 3 in the SDRAMC_LPR. When this mode is activated, all internal voltage generators inside the SDRAM are stopped and all data is lost.

When this mode is enabled, the application must not access to the SDRAM until a new initialization sequence is done (see Section 28.5.1 “SDRAM Device Initialization”).

Refer to Figure 28-8.

Figure 28-8. Deep Power-down Mode Behavior



28.6.6 Scrambling/Unscrambling Function

The external data bus can be scrambled in order to prevent intellectual property data located in off-chip memories from being easily recovered by analyzing data at the package pin level of either microcontroller or memory device.

The scrambling and unscrambling are performed on-the-fly without additional wait states.

The scrambling/unscrambling function can be enabled or disabled by configuring the SDR_SE bit in the SDRAMC OCMS Register (SDRAMC_OCMS). This bit cannot be re-configured as long as the external memory device is powered.

The scrambling method depends on two user-configurable key registers, SDRAMC_OCMS_KEY1 and SDRAMC_OCMS_KEY2 plus a random value depending on device processing characteristics. These key registers are only accessible in Write mode.

The scrambling user key or the seed for key generation must be securely stored in a reliable non-volatile memory in order to recover data from the off-chip memory. Any data scrambled with a given key cannot be recovered if the key is lost.

When multiple chip selects are handled, it is possible to configure the scrambling function per chip select using the OCMS field in the SDRAMC_OCMS registers.

Downloaded from Arrow.com.

28.7 SDRAM Controller (SDRAMC) User Interface

Table 28-7. Register Mapping

Offset	Register	Name	Access	Reset
0x00	SDRAMC Mode Register	SDRAMC_MR	Read/Write	0x00000000
0x04	SDRAMC Refresh Timer Register	SDRAMC_TR	Read/Write	0x00000000
0x08	SDRAMC Configuration Register	SDRAMC_CR	Read/Write	0x852372C0
0x10	SDRAMC Low Power Register	SDRAMC_LPR	Read/Write	0x00000000
0x14	SDRAMC Interrupt Enable Register	SDRAMC_IER	Write-only	–
0x18	SDRAMC Interrupt Disable Register	SDRAMC_IDR	Write-only	–
0x1C	SDRAMC Interrupt Mask Register	SDRAMC_IMR	Read-only	0x00000000
0x20	SDRAMC Interrupt Status Register	SDRAMC_ISR	Read-only	0x00000000
0x24	SDRAMC Memory Device Register	SDRAMC_MDR	Read/Write	0x00000000
0x28	SDRAMC Configuration Register 1	SDRAMC_CFR1	Read/Write	0x00000002
0x2C	SDRAMC OCMS Register	SDRAMC_OCMS	Read/Write	0x00000000
0x30	SDRAMC OCMS KEY1 Register	SDRAMC_OCMS_KEY1	Write-only	–
0x34	SDRAMC OCMS KEY2 Register	SDRAMC_OCMS_KEY2	Write-only	–
0x38–0xFC	Reserved	–	–	–

Note: All unlisted offset values are considered as ‘reserved’.

Downloaded from Arrow.com.

28.7.1 SDRAMC Mode Register

Name: SDRAMC_MR

Address: 0x40084000

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	MODE		

- **MODE: SDRAMC Command Mode**

This field defines the command issued by the SDRAMC when the SDRAM device is accessed.

Value	Name	Description
0	NORMAL	Normal mode. Any access to the SDRAM is decoded normally. To activate this mode, the command must be followed by a write to the SDRAM.
1	NOP	The SDRAMC issues a NOP command when the SDRAM device is accessed regardless of the cycle. To activate this mode, the command must be followed by a write to the SDRAM.
2	ALLBANKS_PRECHARGE	The SDRAMC issues an “All Banks Precharge” command when the SDRAM device is accessed regardless of the cycle. To activate this mode, the command must be followed by a write to the SDRAM.
3	LOAD_MODEREG	The SDRAMC issues a “Load Mode Register” command when the SDRAM device is accessed regardless of the cycle. To activate this mode, the command must be followed by a write to the SDRAM.
4	AUTO_REFRESH	The SDRAMC issues an “Auto-Refresh” Command when the SDRAM device is accessed regardless of the cycle. Previously, an “All Banks Precharge” command must be issued. To activate this mode, the command must be followed by a write to the SDRAM.
5	EXT_LOAD_MODEREG	The SDRAMC issues an “Extended Load Mode Register” command when the SDRAM device is accessed regardless of the cycle. To activate this mode, the “Extended Load Mode Register” command must be followed by a write to the SDRAM. The write in the SDRAM must be done in the appropriate bank; most low-power SDRAM devices use the bank 1.
6	DEEP_POWERDOWN	Deep Power-down mode. Enters Deep Power-down mode.

Downloaded from Arrow.com.

28.7.2 SDRAMC Refresh Timer Register

Name: SDRAMC_TR

Address: 0x40084004

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	COUNT			

7	6	5	4	3	2	1	0
COUNT							

- **COUNT: SDRAMC Refresh Timer Count**

This 12-bit field is loaded into a timer that generates the refresh pulse. Each time the refresh pulse is generated, a refresh burst is initiated. The SDRAM device requires a refresh every 15.625 µs or 7.81 µs. With a 100 MHz frequency, the Refresh Timer Counter Register must be set with the value 1562 (15.625 µs x 100 MHz) or 781 (7.81 µs x 100 MHz).

To refresh the SDRAM device, this 12-bit field must be written. If this condition is not satisfied, no refresh command is issued and no refresh of the SDRAM device is carried out.

Downloaded from Arrow.com.

28.7.3 SDRAMC Configuration Register

Name: SDRAMC_CR

Address: 0x40084008

Access: Read/Write

31	30	29	28	27	26	25	24
TXSR				TRAS			
23	22	21	20	19	18	17	16
TRCD				TRP			
15	14	13	12	11	10	9	8
TRC_TRFC				TWR			
7	6	5	4	3	2	1	0
DBW	CAS		NB	NR		NC	

Warning: Bit 7 (DBW) must always be set when programming the SDRAMC_CR.

- **NC: Number of Column Bits**

Reset value is 8 column bits.

Value	Name	Description
0	COL8	8 column bits
1	COL9	9 column bits
2	COL10	10 column bits
3	COL11	11 column bits

- **NR: Number of Row Bits**

Reset value is 11 row bits.

Value	Name	Description
0	ROW11	11 row bits
1	ROW12	12 row bits
2	ROW13	13 row bits
3	–	Reserved

- **NB: Number of Banks**

Reset value is two banks.

Value	Name	Description
0	BANK2	2 banks
1	BANK4	4 banks

Downloaded from Arrow.com.

- **CAS: CAS Latency**

Reset value is two cycles. In the SDRAMC, only a CAS latency of one, two and three cycles are managed.

Value	Name	Description
0	–	Reserved
1	LATENCY1	1 cycle latency
2	LATENCY2	2 cycle latency
3	LATENCY3	3 cycle latency

- **DBW: Data Bus Width**

Reset value is 16 bits.

This bit defines the Data Bus Width, which is 16 bits. It must be set to 1.

- **TWR: Write Recovery Delay**

Reset value is two cycles.

This field defines the Write Recovery Time in number of cycles. Number of cycles is between 0 and 15.

- **TRC_TRFC: Row Cycle Delay and Row Refresh Cycle**

Reset value is seven cycles.

This field defines two timings:

– the delay (t_{RFC}) between two Refresh commands and between a Refresh command and an Activate command

– and the delay (t_{RC}) between two Active commands in number of cycles.

The number of cycles is between 0 and 15. The end user must program max {t_{RC}, t_{RFC}}.

- **TRP: Row Precharge Delay**

Reset value is three cycles.

This field defines the delay between a Precharge Command and another Command in number of cycles. Number of cycles is between 0 and 15.

- **TRCD: Row to Column Delay**

Reset value is two cycles.

This field defines the delay between an Activate Command and a Read/Write Command in number of cycles. Number of cycles is between 0 and 15.

- **TRAS: Active to Precharge Delay**

Reset value is five cycles.

This field defines the delay between an Activate Command and a Precharge Command in number of cycles. Number of cycles is between 0 and 15.

- **TXSR: Exit Self-Refresh to Active Delay**

Reset value is eight cycles.

This field defines the delay between SCKE set high and an Activate Command in number of cycles. Number of cycles is between 0 and 15.

Downloaded from Arrow.com.

28.7.4 SDRAMC Low Power Register

Name: SDRAMC_LPR

Address: 0x40084010

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	TIMEOUT		DS		TCSR	

7	6	5	4	3	2	1	0
–	PASR			–	–	LPCB	

- **LPCB: Low-power Configuration Bits**

Value	Name	Description
0	DISABLED	The low-power feature is inhibited: no Power-down, Self-refresh or Deep Power-down command is issued to the SDRAM device.
1	SELF_REFRESH	The SDRAMC issues a Self-refresh command to the SDRAM device, the SDCK clock is deactivated and the SDCKE signal is set low. The SDRAM device leaves the Self-refresh mode when accessed and enters it after the access.
2	POWER_DOWN	The SDRAMC issues a Power-down Command to the SDRAM device after each access, the SDCKE signal is set to low. The SDRAM device leaves the Power-down mode when accessed and enters it after the access.
3	DEEP_POWER_DOWN	The SDRAMC issues a Deep Power-down command to the SDRAM device. This mode is unique to low-power SDRAM.

- **PASR: Partial Array Self-refresh (only for low-power SDRAM)**

PASR parameter is transmitted to the SDRAM during initialization to specify whether only one quarter, one half or all banks of the SDRAM array are enabled. Disabled banks are not refreshed in Self-refresh mode. This parameter must be set according to the SDRAM device specification.

After initialization, as soon as the PASR field is modified and Self-refresh mode is activated, the Extended Mode Register is accessed automatically and PASR bits are updated before entry in Self-refresh mode. This feature is not supported when SDRAMC shares an external bus with another controller.

- **TCSR: Temperature Compensated Self-Refresh (only for low-power SDRAM)**

TCSR parameter is transmitted to the SDRAM during initialization to set the refresh interval during Self-refresh mode depending on the temperature of the low-power SDRAM. This parameter must be set according to the SDRAM device specification.

After initialization, as soon as the TCSR field is modified and Self-refresh mode is activated, the Extended Mode Register is accessed automatically and TCSR bits are updated before entry in Self-refresh mode. This feature is not supported when SDRAMC shares an external bus with another controller.

- **DS: Drive Strength (only for low-power SDRAM)**

DS parameter is transmitted to the SDRAM during initialization to select the SDRAM strength of data output. This parameter must be set according to the SDRAM device specification.

Downloaded from Arrow.com.

After initialization, as soon as the DS field is modified and Self-refresh mode is activated, the Extended Mode Register is accessed automatically and DS bits are updated before entry in Self-refresh mode. This feature is not supported when SDRAMC shares an external bus with another controller.

- **TIMEOUT: Time to Define When Low-power Mode Is Enabled**

Value	Name	Description
0	LP_LAST_XFER	The SDRAMC activates the SDRAM Low-power mode immediately after the end of the last transfer.
1	LP_LAST_XFER_64	The SDRAMC activates the SDRAM Low-power mode 64 clock cycles after the end of the last transfer.
2	LP_LAST_XFER_128	The SDRAMC activates the SDRAM Low-power mode 128 clock cycles after the end of the last transfer.
3	–	Reserved

Downloaded from Arrow.com.

28.7.5 SDRAMC Interrupt Enable Register

Name: SDRAMC_IER

Address: 0x40084014

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	RES

- **RES: Refresh Error Status**

0: No effect.

1: Enables the refresh error interrupt.

Downloaded from Arrow.com.

28.7.6 SDRAMC Interrupt Disable Register

Name: SDRAMC_IDR

Address: 0x40084018

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	RES

- **RES: Refresh Error Status**

0: No effect.

1: Disables the refresh error interrupt.

Downloaded from Arrow.com.

28.7.7 SDRAMC Interrupt Mask Register

Name: SDRAMC_IMR

Address: 0x4008401C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	RES

- **RES: Refresh Error Status**

0: The refresh error interrupt is disabled.

1: The refresh error interrupt is enabled.

Downloaded from Arrow.com.

28.7.8 SDRAMC Interrupt Status Register

Name: SDRAMC_ISR

Address: 0x40084020

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	RES

- **RES: Refresh Error Status (cleared on read)**

0: No refresh error has been detected since the register was last read.

1: A refresh error has been detected since the register was last read.

Downloaded from Arrow.com.

28.7.9 SDRAMC Memory Device Register

Name: SDRAMC_MDR

Address: 0x40084024

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	MD	

- **MD: Memory Device Type**

Value	Name	Description
0	SDRAM	SDRAM
1	LPSDRAM	Low-power SDRAM
2	–	Reserved
3	–	Reserved

Downloaded from Arrow.com.

28.7.10 SDRAMC Configuration Register 1

Name: SDRAMC_CFR1

Address: 0x40084028

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	UNAL

7	6	5	4	3	2	1	0
–	–	–	–	TMRD			

- **TMRD: Load Mode Register Command to Active or Refresh Command**

Reset value is 2 cycles.

This field defines the delay between a “Load Mode Register” command and an active or refresh command in number of cycles. Number of cycles is between 0 and 15.

- **UNAL: Support Unaligned Access**

Value	Name	Description
0	UNSUPPORTED	Unaligned access is not supported.
1	SUPPORTED	Unaligned access is supported.

This mode is enabled with masters which have an AXI interface.

Downloaded from Arrow.com.

28.7.11 SDRAMC OCMS Register

Name: SDRAMC_OCMS

Address: 0x4008402C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	SDR_SE

- **SDR_SE: SDRAM Memory Controller Scrambling Enable**

0: Disables off-chip scrambling for SDR-SDRAM access.

1: Enables off-chip scrambling for SDR-SDRAM access.

Downloaded from Arrow.com.

28.7.12 SDRAMC OCMS KEY1 Register

Name: SDRAMC_OCMS_KEY1

Address: 0x40084030

Access: Write-once

31	30	29	28	27	26	25	24
KEY1							

23	22	21	20	19	18	17	16
KEY1							

15	14	13	12	11	10	9	8
KEY1							

7	6	5	4	3	2	1	0
KEY1							

- **KEY1: Off-chip Memory Scrambling (OCMS) Key Part 1**

When off-chip memory scrambling is enabled, the data scrambling depends on KEY1 and KEY2 values.

Downloaded from Arrow.com.

28.7.13 SDRAMC OCMS KEY2 Register

Name: SDRAMC_OCMS_KEY2

Address: 0x40084034

Access: Write-once

31	30	29	28	27	26	25	24
KEY2							

23	22	21	20	19	18	17	16
KEY2							

15	14	13	12	11	10	9	8
KEY2							

7	6	5	4	3	2	1	0
KEY2							

- **KEY2: Off-chip Memory Scrambling (OCMS) Key Part 2**

When off-chip memory scrambling is enabled, the data scrambling depends on KEY1 and KEY2 values.

Downloaded from Arrow.com.

29. General Purpose Backup Registers (GPBR)

29.1 Description

The System Controller embeds 256 bits of General Purpose Backup registers organized as 8 32-bit registers.

It is possible to generate an immediate clear of the content of General Purpose Backup registers 0 to 3 (first half) if a Low-power Debounce event is detected on one of the wakeup pins, WKUP0 or WKUP1. The content of the other General Purpose Backup registers (second half) remains unchanged.

The Supply Controller module must be programmed accordingly. In the register SUPC_WUMR in the Supply Controller module, LPDBCCLR, LPDBCEN0 and/or LPDBCEN1 bit must be configured to 1 and LPDBC must be other than 0.

If a Tamper event has been detected, it is not possible to write to the General Purpose Backup registers while the LPDBCS0 or LPDBCS1 flags are not cleared in the Supply Controller Status Register (SUPC_SR).

29.2 Embedded Characteristics

- 256 bits of General Purpose Backup Registers

Downloaded from Arrow.com.

29.3 General Purpose Backup Registers (GPBR) User Interface

Table 29-1. Register Mapping

Offset	Register	Name	Access	Reset
0x0	General Purpose Backup Register 0	SYS_GPBR0	Read/Write	0x00000000
...	...	...	...	...
0x1C	General Purpose Backup Register 7	SYS_GPBR7	Read/Write	0x00000000

Downloaded from Arrow.com.

29.3.1 General Purpose Backup Register x

Name: SYS_GPBRx

Address: 0x400E1890

Access: Read/Write

31	30	29	28	27	26	25	24
GPBR_VALUE							

23	22	21	20	19	18	17	16
GPBR_VALUE							

15	14	13	12	11	10	9	8
GPBR_VALUE							

7	6	5	4	3	2	1	0
GPBR_VALUE							

These registers are reset at first power-up and on each loss of VDDIO.

- **GPBR_VALUE: Value of GPBR x**

If a Tamper event has been detected, it is not possible to write GPBR_VALUE as long as the LPDBCS0 or LPDBCS1 flag has not been cleared in the Supply Controller Status Register (SUPC_SR).

Downloaded from Arrow.com.

30. Clock Generator

30.1 Description

The Clock Generator user interface is embedded within the Power Management Controller and is described in Section 31.20 ”Power Management Controller (PMC) User Interface”. However, the Clock Generator registers are named CKGR_.

30.2 Embedded Characteristics

The Clock Generator is made up of:

- A low-power 32.768 kHz crystal oscillator with Bypass mode
- A low-power embedded 32 kHz (typical) RC oscillator
- A 3 to 20 MHz crystal or ceramic resonator-based oscillator with Bypass mode
- An embedded RC oscillator. Three output frequencies can be selected: 4/8/12 MHz. By default 4 MHz is selected. 8 MHz and 12 MHz are factory-trimmed.
- A 480 MHz UTMI PLL, providing a clock for the USB High-speed Controller
- A 160 to 500 MHz programmable PLL (input from 8 to 32 MHz)

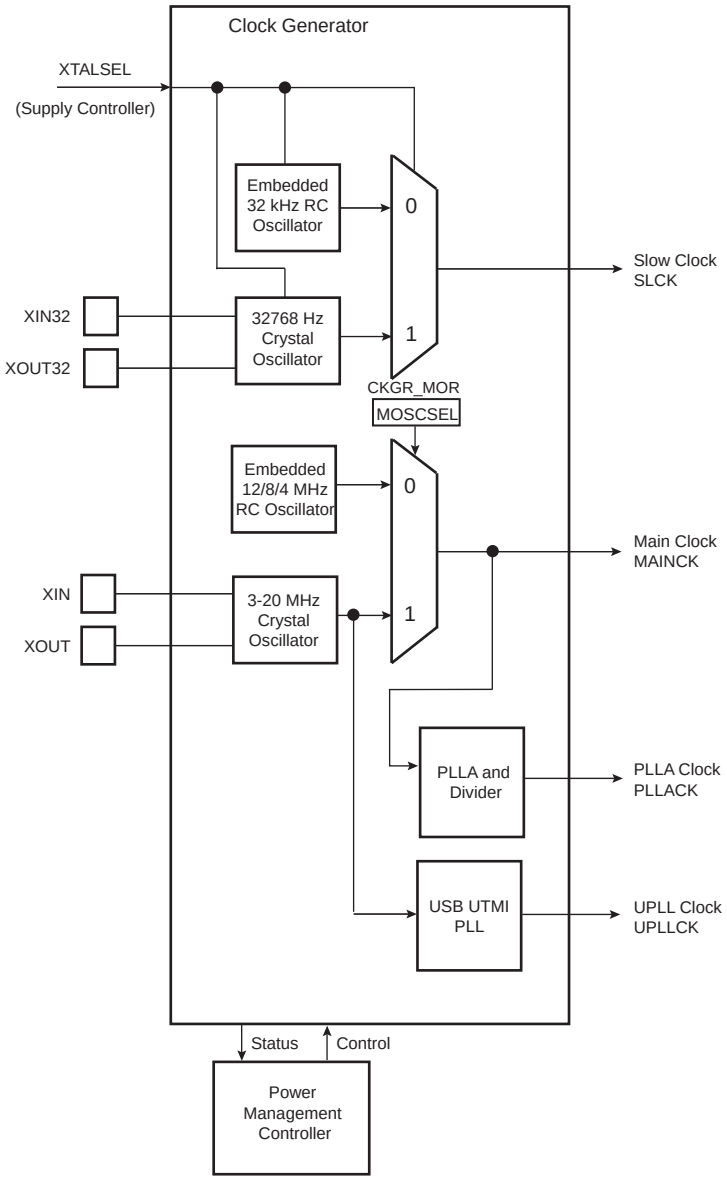
It provides the following clocks:

- SLCK, the slow clock, which is the only permanent clock within the system.
- MAINCK is the output of the main clock oscillator selection: either the crystal or ceramic resonator-based oscillator or 4/8/12 MHz RC oscillator.
- PLLACK is the output of the divider and 160 to 500 MHz programmable PLL (PLLA)
- UPLLCK is the output of the 480 MHz UTMI PLL (UPLL)

Downloaded from Arrow.com.

30.3 Block Diagram

Figure 30-1. Clock Generator Block Diagram



Downloaded from Arrow.com.

30.4 Slow Clock

The Supply Controller embeds a slow clock generator that is supplied with the VDDIO power supply. As soon as VDDIO is supplied, both the crystal oscillator and the embedded RC oscillator are powered up, but only the embedded 32 kHz (typical) RC oscillator is enabled. This allows the slow clock to be valid in a short time (about 100 µs).

The slow clock is generated either by the 32.768 kHz crystal oscillator or by the embedded 32 kHz (typical) RC oscillator.

The selection is made via the XTALSEL bit in the Supply Controller Control register (SUPC_CR).

30.4.1 Embedded 32 kHz (typical) RC Oscillator

By default, the embedded 32 kHz (typical) RC oscillator is enabled and selected. The user has to take into account the possible drifts of this oscillator. Refer to section Section 56.2 ”DC Characteristics”.

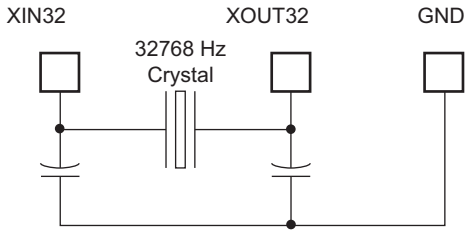
This oscillator is disabled by clearing the SUPC_CR.XTALSEL.

30.4.2 32.768 kHz Crystal Oscillator

The Clock Generator integrates a low-power 32.768 kHz oscillator. To use this oscillator, the XIN32 and XOUT32 pins must be connected to a 32.768 kHz crystal. Two external capacitors must be wired as shown in Figure 30-2. More details are given in the section “DC Characteristics”.

Note that the user is not obliged to use the 32.768 kHz crystal oscillator and can use the 32 kHz (typical) RC oscillator instead.

Figure 30-2. Typical 32.768 kHz Crystal Oscillator Connection



The 32.768 kHz crystal oscillator provides a more accurate frequency than the 32 kHz (typical) RC oscillator.

To select the 32.768 kHz crystal oscillator as the source of the slow clock, the bit SUPC_CR.XTALSEL must be set. This results in a sequence which first configures the PIO lines multiplexed with XIN32 and XOUT32 to be driven by the crystal oscillator, then enables the 32.768 kHz crystal oscillator and then disables the 32 kHz (typical) RC oscillator to save power. The switch of the slow clock source is glitch-free.

Reverting to the 32 kHz (typical) RC oscillator is only possible by shutting down the VDDIO power supply. If the user does not need the 32.768 kHz crystal oscillator, the XIN32 and XOUT32 pins can be left unconnected since by default the XIN32 and XOUT32 system I/O pins are in PIO input mode with pull-up after reset.

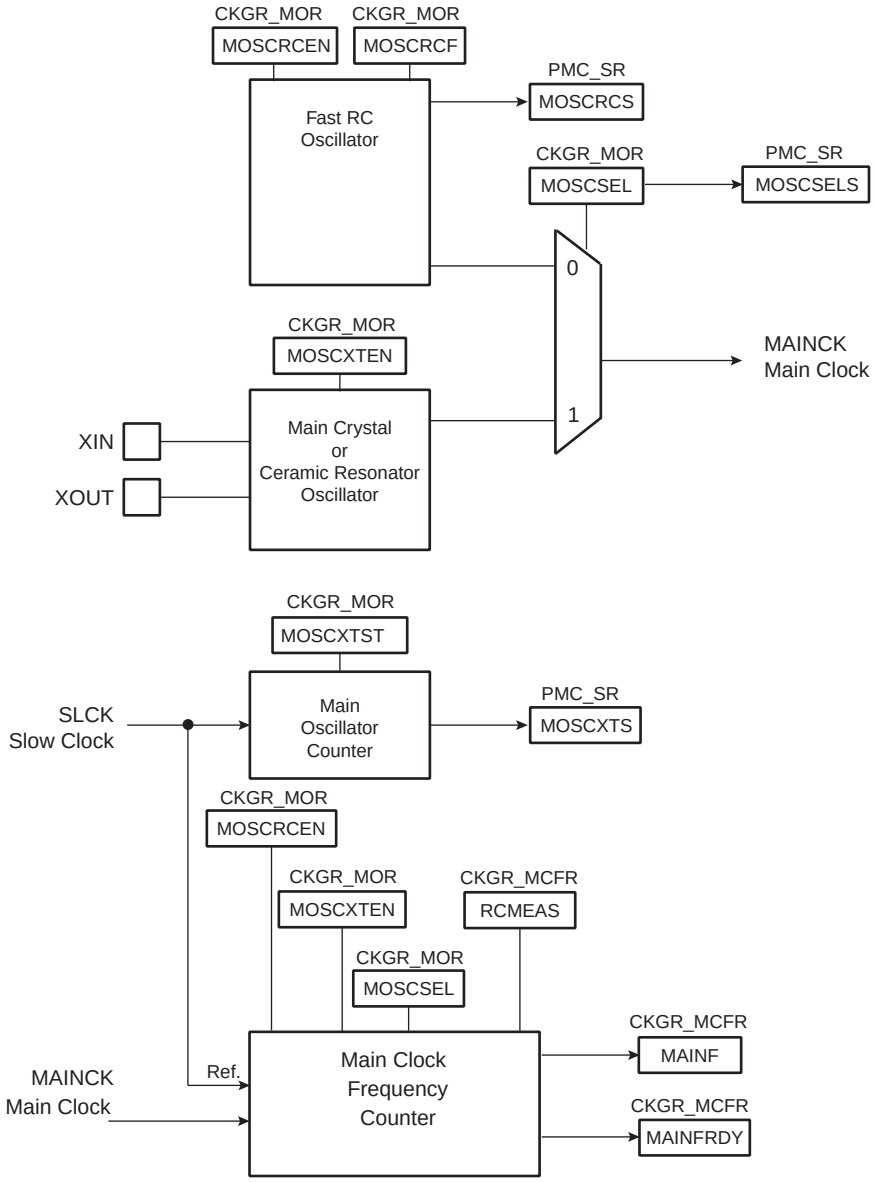
The user can also set the 32.768 kHz crystal oscillator in Bypass mode instead of connecting a crystal. In this case, the user must provide the external clock signal on XIN32. The input characteristics of the XIN32 pin are given in the section “Electrical Characteristics”. To enter Bypass mode, the OSCBYPASS bit of the Supply Controller Mode register (SUPC_MR) must be set prior to setting SUPC_CR.XTALSEL.

30.5 Main Clock

Figure 30-3 shows the main clock block diagram.

Downloaded from Arrow.com.

Figure 30-3. Main Clock Block Diagram



The main clock has two sources:

- A 4/8/12 MHz RC oscillator with a fast start-up time and that is selected by default to start the system
- A 3 to 20 MHz crystal or ceramic resonator-based oscillator which can be bypassed. Refer to Section 30.5.5 ”Bypassing the Main Crystal Oscillator”.

30.5.1 Embedded 4/8/12 MHz RC Oscillator

After reset, the 4/8/12 MHz RC oscillator is enabled with the 4 MHz frequency selected. This oscillator is selected as the source of MAINCK. MAINCK is the default clock selected to start the system.

Only the 8/12 MHz RC oscillator frequencies are calibrated in production.

Refer to Section 56.2 ”DC Characteristics”.

The software can disable or enable the 4/8/12 MHz RC oscillator with the MOSCRCEN bit in the Clock Generator Main Oscillator register (CKGR_MOR).

Downloaded from Arrow.com.

The output frequency of the RC oscillator can be selected among 4, 8 or 12 MHz. Selection is done by configuring the field MOSCRCF in CKGR_MOR. When changing the frequency selection, the MOSCRCS bit in the Power Management Controller Status register (PMC_SR) is automatically cleared and MAINCK is stopped until the oscillator is stabilized. Once the oscillator is stabilized, the main clock restarts and PMC_SR.MOSCRCS is set.

When disabling the main clock by clearing the CKGR_MOR.MOSCRCEN bit, the PMC_SR.MOSCRCS bit is automatically cleared, indicating the main clock is off.

Setting the MOSCRCS bit in the Power Management Controller Interrupt Enable register (PMC_IER) triggers an interrupt to the processor.

When the main clock (MAINCK) is not used to drive the processor and the frequency monitor (SLCK is used instead), it is recommended to disable the 4/8/12 MHz RC oscillator and 3 to 20 MHz crystal oscillator.

The user can adjust the value of the fast RC oscillator frequency by modifying the trimming values done in production by Atmel on 8 MHz and 12 MHz. Refer to Section 30.5.2 "4/8/12 MHz RC Oscillator Clock Frequency Adjustment". The values stored in the Flash cannot be erased by a Flash erase command or by the ERASE pin. Values written by the user application in PMC_OCR are reset after each power-up or peripheral reset.

30.5.2 4/8/12 MHz RC Oscillator Clock Frequency Adjustment

The user can adjust the 4/8/12 MHz RC oscillator frequency in PMC_OCR. By default, SEL4/8/12 are cleared, so the RC oscillator will be driven with Flash calibration bits which are programmed during chip production.

The user can adjust the trimming of the 4/8/12 MHz fast RC oscillator through this register to obtain more accurate frequency and to compensate derating factors such as temperature and voltage.

In order to calibrate the oscillator lower frequency, SEL4 must be set to '1' and a valid frequency value must be configured in CAL4. Likewise, SEL8/12 must be set to '1' and a trim value must be configured in CAL8/12 in order to adjust the other frequencies of the oscillator.

It is possible to adjust the oscillator frequency while operating from this clock. For example, when running on lowest frequency it is possible to change the CAL4 value if SEL4 is set in PMC_OCR.

At any time, it is possible to restart a measurement of the frequency of the selected clock via the RCMEAS bit in Main Clock Frequency register (CKGR_MCFR). Thus, when CKGR_MCFR.MAINFRDY flag reads 1, another read access on CKGR_MCFR provides an image of the frequency of the main clock on CKGR_MCFR.MAINF. The software can calculate the error with an expected frequency and correct the CAL4, CAL8 and CAL12 fields accordingly. This may be used to compensate frequency drift due to derating factors such as temperature and/or voltage.

30.5.3 3 to 20 MHz Crystal or Ceramic Resonator-based Oscillator

After reset, the 3 to 20 MHz crystal or ceramic resonator-based oscillator is disabled and is not selected as the source of the main clock.

As the source of MAINCK, the 3 to 20 MHz crystal or ceramic resonator-based oscillator provides a very precise frequency. The software enables or disables this oscillator in order to reduce power consumption via CKGR_MOR.MOSCXTEN.

When disabling this oscillator by clearing the CKGR_MOR.MOSCXTEN, PMC_SR.MOSCXTS is automatically cleared, indicating the 3 to 20 MHz crystal oscillator is off.

When enabling this oscillator, the user must initiate the start-up time counter. The start-up time depends on the characteristics of the external device connected to this oscillator.

When CKGR_MOR.MOSCXTEN and CKGR_MOR.MOSCXTST are written to enable this oscillator, the XIN and XOUT pins are automatically switched into Oscillator mode. PMC_SR.MOSCXTS is cleared and the counter starts counting down on the slow clock divided by 8 from the CKGR_MOR.MOSCXTST value. Since the CKGR_MOR.MOSCXTST value is coded with 8 bits, the maximum start-up time is about 62 ms.

Downloaded from Arrow.com.

When the start-up time counter reaches 0, PMC_SR.MOSCXTS is set, indicating that the 3 to 20 MHz crystal oscillator is stabilized. Setting the MOSCXTS bit in the Interrupt Mask register (PMC_IMR) can trigger an interrupt to the processor.

30.5.4 Main Clock Source Selection

The user can select the source of the main clock from either the 4/8/12 MHz fast RC oscillator, the 3 to 20 MHz crystal oscillator or the ceramic resonator-based oscillator.

The advantage of the 4/8/12 MHz fast RC oscillator is its fast start-up time. By default, this oscillator is selected to start the system.

The 4/8/12 MHz fast RC oscillator must be selected when entering Wait mode.

The advantage of the 3 to 20 MHz crystal oscillator or ceramic resonator-based oscillator is the high level of accuracy provided.

The selection of the oscillator is made by writing CKGR_MOR.MOSCSEL. The switch of the main clock source is glitch-free, so there is no need to run out of SLCK, PLLACK or PLLBCK or UPLLCK in order to change the selection. PMC_SR.MOSCSELS indicates when the switch sequence is done.

Setting PMC_IMR.MOSCSELS triggers an interrupt to the processor.

Enabling the 4/8/12 MHz RC oscillator (MOSCRCEN = 1) and changing its frequency (MOSCCRF) at the same time is not allowed.

This oscillator must be enabled first and its frequency changed in a second step.

30.5.5 Bypassing the Main Crystal Oscillator

Prior to bypassing the 3 to 20 MHz crystal oscillator, the external clock frequency provided on the XIN pin must be stable and within the values specified in the XIN Clock characteristics in the section “Electrical Characteristics”.

The sequence is as follows:

1. Ensure that an external clock is connected on XIN.
2. Enable the bypass by setting CKGR_MOR.MOSCXTBY.
3. Disable the 3 to 20 MHz oscillator by clearing the bit CKGR_MOR.MOSCXTEN.

30.5.6 Main Clock Frequency Counter

The frequency counter is managed by CKGR_MCFR.

During the measurement period, the frequency counter increments at the speed of the clock defined by the bit CKGR_MCFR.CCSS.

A measurement is started in the following cases:

- When the RCMEAS bit of CKGR_MCFR is written to 1.
- When the 4/8/12 MHz RC oscillator is selected as the source of main clock and when this oscillator becomes stable (i.e., when the MOSCRCS bit is set)
- When the 3 to 20 MHz crystal or ceramic resonator-based oscillator is selected as the source of main clock and when this oscillator becomes stable (i.e., when the MOSCXTS bit is set)
- When the main clock source selection is modified

The measurement period ends at the 16th falling edge of slow clock, the MAINFRDY bit in CKGR_MCFR is set and the counter stops counting. Its value can be read in the MAINF field of CKGR_MCFR and gives the number of clock cycles during 16 periods of slow clock, so that the frequency of the 4/8/12 MHz RC oscillator or 3 to 20 MHz crystal or ceramic resonator-based oscillator can be determined.

Downloaded from Arrow.com.

30.5.7 Switching Main Clock between the RC Oscillator and Crystal Oscillator

When switching the source of the main clock between the RC oscillator and the crystal oscillator, both oscillators must be enabled. After completion of the switch, the unused oscillator can be disabled.

If switching to the crystal oscillator, follow the programming sequence below to ensure that the oscillator is present and that its frequency is valid:

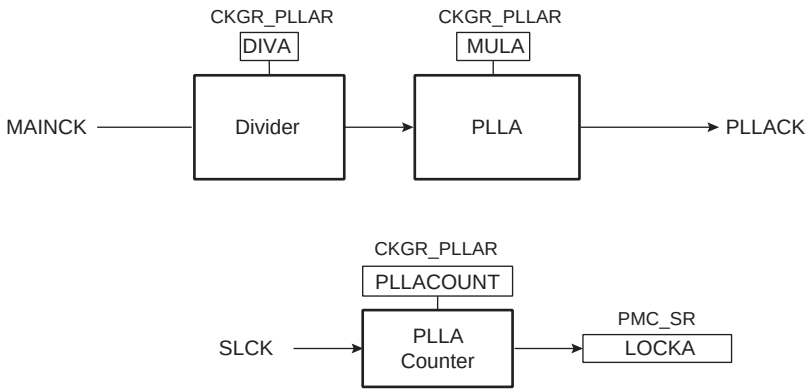
1. Enable the crystal oscillator by setting CKGR_MOR.MOSCXTEN. Configure the CKGR_MOR. MOSCXTST field with the crystal oscillator start-up time as defined in the section “Electrical Characteristics”.
2. Wait for PMC_SR.MOSCXTS flag to rise, indicating the end of a start-up period of the crystal oscillator.
3. Select the crystal oscillator as the source clock of the frequency meter by setting CKGR_MCFR.CCSS.
4. Initiate a frequency measurement by setting CKGR_MCFR.RCMEAS.
5. Read CKGR_MCFR.MAINFRDY until its value equals 1.
6. Read CKGR_MCFR.MAINF and compute the value of the crystal frequency.
7. If the MAINF value is valid, the main clock can be switched to the 3 to 20 MHz crystal oscillator.

30.6 Divider and PLL Block

The device features one divider/one PLL block that permits a wide range of frequencies to be selected on either the master clock, the processor clock or the programmable clock outputs. Additionally, they provide a 48 MHz signal to the embedded USB device port regardless of the frequency of the main clock.

Figure 30-4 shows the block diagram of the dividers and PLL blocks.

Figure 30-4. Divider and PLL Block Diagram



30.6.1 Divider and Phase Lock Loop Programming

The divider can be set between 1 and 255 in steps of 1. When a divider field (DIV) is cleared, the output of the corresponding divider and the PLL output is a continuous signal at level 0. On reset, each DIV field is cleared, thus the corresponding PLL input clock is stuck at 0.

The PLL (PLLA) allows multiplication of the divider’s outputs. The PLL clock signal has a frequency that depends on the respective source signal frequency and on the parameters DIV (DIVA) and MUL (MULA). The factor applied to the source signal frequency is (MUL + 1)/DIV. When MUL is written to ‘0’ or DIV=0, the PLL is disabled and its power consumption is saved. Note that there is a delay of two SLCK clock cycles between the disable command and the real disable of the PLL. Re-enabling the PLL can be performed by writing a value higher than 0 in the MUL field and DIV higher than 0.

Whenever the PLL is re-enabled or one of its parameters is changed, the LOCK (LOCKA) bit in PMC_SR is automatically cleared. The values written in the PLLCOUNT field (PLLACOUNT) in CKGR_PLLR (CKGR_PLLAR) are loaded in the PLL counter. The PLL counter then decrements at the speed of SLCK until it reaches 0. At this

Downloaded from Arrow.com.

time, PMC_SR.LOCK is set and can trigger an interrupt to the processor. The user has to load the number of slow clock cycles required to cover the PLL transient time into the PLLCOUNT field.

To avoid programming the PLL with a multiplication factor that is too high, the user can saturate the multiplication factor value sent to the PLL by setting the PLLA_MMAX field in PMC_PMMR.

It is prohibited to change the 4/8/12 MHz fast RC oscillator or the main oscillator selection in CKGR_MOR while the master clock source is the PLL and the PLL reference clock is the fast RC oscillator.

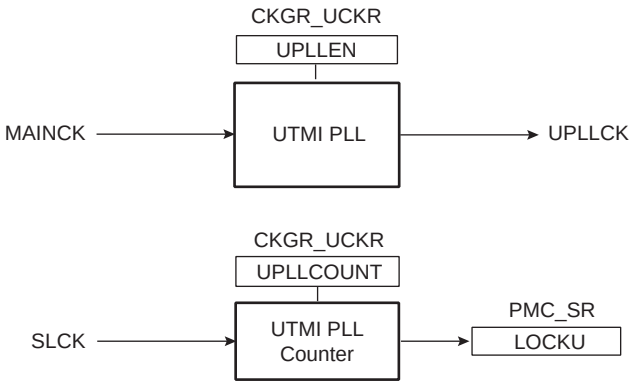
The user must:

1. Switch on the main RC oscillator by writing a '1' to PMC_MCKR.CSS.
2. Change the frequency (MOSCRCF) or oscillator selection (MOSCSEL) in CKGR_MOR.
3. Wait for MOSCRCS (if frequency changes) or MOSCSELS (if oscillator selection changes) in PMC_SR.
4. Disable and then enable the PLL.
5. Wait for the LOCK flag in PMC_SR.
6. Switch back to the PLL by writing the appropriate value to PMC_MCKR.CSS.

30.7 UTMI Phase Lock Loop Programming

The source clock of the UTMI PLL is the 3 to 20 MHz crystal oscillator.

Figure 30-5. UTMI PLL Block Diagram



Whenever the UTMI PLL is enabled by writing UPLLEN in UTMI Clock register (CKGR_UCKR), the LOCKU bit in PMC_SR is automatically cleared. The values written in the PLLCOUNT field in CKGR_UCKR are loaded in the UTMI PLL counter. The UTMI PLL counter then decrements at the speed of the slow clock divided by 8 until it reaches 0. At this time, the LOCKU bit is set in PMC_SR and can trigger an interrupt to the processor. The user has to load the number of slow clock cycles required to cover the UTMI PLL transient time into the PLLCOUNT field.

Downloaded from Arrow.com.

31. Power Management Controller (PMC)

31.1 Description

The Power Management Controller (PMC) optimizes power consumption by controlling all system and user peripheral clocks. The PMC enables/disables the clock inputs to many of the peripherals and the Cortex-M7 processor.

The Supply Controller selects either the embedded 32 kHz RC oscillator or the 32.768 kHz crystal oscillator. The unused oscillator is disabled automatically so that power consumption is optimized.

By default, at startup, the chip runs out of the master clock using the 4/8/12 MHz RC oscillator running at 4 MHz.

The user can trim the 4/8/12 MHz RC oscillator frequencies by software.

31.2 Embedded Characteristics

The Power Management Controller provides the following clocks:

- MCK, the Master Clock, programmable from a few hundred Hz to the maximum operating frequency of the device. It is available to the modules running permanently, such as the Enhanced Embedded Flash Controller.
- Processor Clock (HCLK), automatically switched off when entering the processor in Sleep Mode.
- Free-running processor Clock (FCLK)
- the Cortex-M7 SysTick external clock
- USB Clock (USBCK), required by USB Device Port operations.
- Peripheral Clocks, provided to the embedded peripherals (USART, SPI, TWI, TC, etc.) and independently controllable.
- Programmable Clock Outputs (PCKx), selected from the clock generator outputs to drive the device PCK pins.
- Clock sources independent of MCK and HCLK, provided by internal PCKx for USART, UART, and TC
- Generic Clock (GCLK) independent of MCK and HCLK
- Embedded Trace Macrocell (ETM) and CAN Clocks, provided by internal PCKx clocks

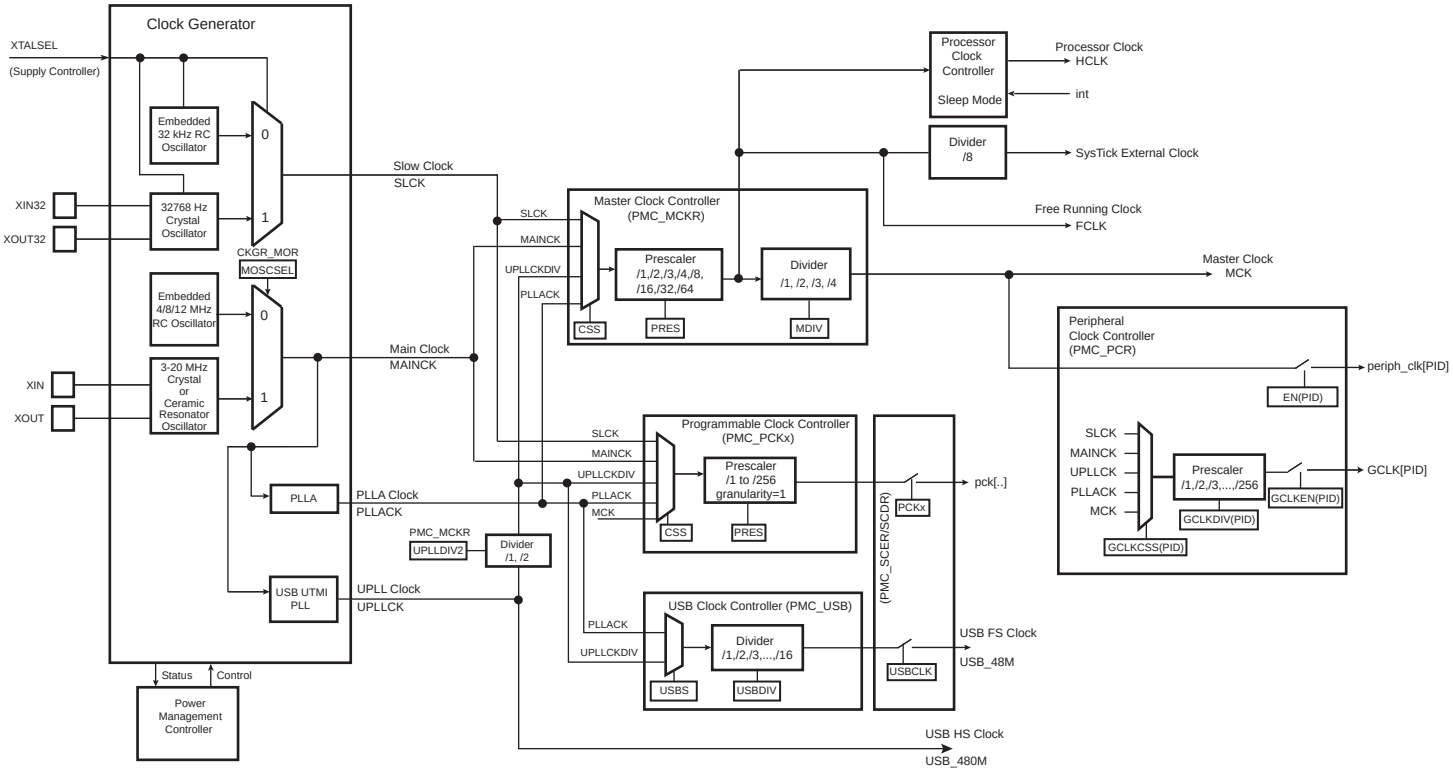
The Power Management Controller also provides the following features on clocks:

- A 3 to 20 MHz crystal oscillator clock failure detector.
- A 32.768 kHz crystal oscillator frequency monitor.
- A frequency counter on 3 to 20 MHz crystal oscillator or 4/8/12 MHz RC oscillator
- An on-the-fly adjustable 4/8/12 MHz RC oscillator frequency

Downloaded from Arrow.com.

31.3 Block Diagram

Figure 31-1. General Clock Block Diagram



31.4 Master Clock Controller

The Master Clock Controller provides selection and division of the master clock (MCK). MCK is the source clock of the peripheral clocks.

MCK is selected from one of the clocks provided by the Clock Generator. Selecting the slow clock (SLCK) provides a slow clock signal to the whole device. Selecting the main clock saves power consumption of the PLLs.

The Master Clock Controller is made up of a clock selector and a prescaler.

MCK is selected by configuring PMC_MCKR.CSS. The prescaler supports the division by a power of 2 of the selected clock between 1 and 64, and the division by 3. The prescaler is configured using PMC_MCKR.PRES.

Each time PMC_MCKR is written to define a new master clock, the MCKRDY bit is cleared in PMC_SR. It reads ‘0’ until the master clock is established. Then, the MCKRDY bit is set and can trigger an interrupt to the processor. This feature is useful when switching from a high-speed clock to a lower one to inform the software when the change is completed.

Downloaded from Arrow.com.

31.5 Processor Clock Controller

The PMC features a Processor Clock (HCLK) Controller that implements the processor Sleep mode. HCLK can be disabled by executing the WFI (WaitForInterrupt) or the WFE (WaitForEvent) processor instruction while the LPM bit is at 0 in the PMC Fast Startup Mode register (PMC_FSMR).

HCLK is enabled after a reset and is automatically re-enabled by any enabled interrupt. The processor Sleep mode is entered by disabling the processor clock, which is automatically re-enabled by any enabled fast or normal interrupt, or by the reset of the product.

When processor Sleep mode is entered, the current instruction is finished before the clock is stopped, but this does not prevent data transfers from other masters of the system bus.

31.6 SysTick External Clock

When the processor selects the SysTick external clock, the calibration value is fixed to 37500. This allows the generation of a time base of 1 ms with the SysTick clock at the maximum frequency on MCK divided by 8.

Refer to Section 14. ”ARM Cortex-M7 Processor” for details on selecting the SysTick external clock.

31.7 USB Clock Controller

The user can select the PLLA or the UPLL output as the USB source clock by writing the USBS bit in PMC_USB. If using the USB, the user must program the PLL to generate an appropriate frequency depending on the USBDIV bit in the USB Clock register (PMC_USB).

When PMC_SR.LOCKA and PMC_SR.LOCKU are set to ‘1’, the PLLA and UPLL are stable. Then, the USB FS clock can be enabled by setting the USBCLK bit in the System Clock Enable register (PMC_SCER). To save power on this peripheral when not used, the user can set the USBCLK bit in the System Clock Disable register (PMC_SCDR). The USBCLK bit in the System Clock Status register (PMC_SCSR) gives the status of this clock. The USB port requires both the USB clock signal and the peripheral clock. The USB peripheral clock is controlled by means of the Master Clock Controller.

31.8 Peripheral Clock Controller

The PMC controls the clocks of the embedded peripherals by means of the Peripheral Control register (PMC_PCR). With this register, the user can enable and disable the clock of each peripheral.

This register is also used to enable, disable and configure the GCLK of I2SC0 and I2SC1. GCLK is independent of MCK and HCLK.

To configure the clocks of a peripheral, PMC_PCR.CMD must be written to ‘1’ and PMC_PCR.PID must be written with the index of the corresponding peripheral. All other configuration fields must be correctly set.

To read the current clock configuration of a peripheral, PMC_PCR.CMD must be written to ‘0’ and PMC_PCR.PID must be written with the index of the corresponding peripheral regardless of the values of other fields. This write does not modify the configuration of the peripheral. The PMC_PCR register can then be read to know the configuration status of the corresponding PID.

The user can also enable and disable these clocks by writing Peripheral Clock Enable (PMC_PCERx) and Peripheral Clock Disable (PMC_PCDRx) registers. The status of the peripheral clock activity can be read in the Peripheral Clock Status registers (PMC_PCSRx).

When the peripheral clock is disabled, it is immediately stopped. The peripheral clocks are automatically disabled after a reset.

To stop a peripheral clock, it is recommended that the system software wait until the peripheral has executed its last programmed operation before disabling the clock. This is to avoid data corruption or erroneous behavior of the system.

Downloaded from Arrow.com.

The bit number in PMC_PCERx, PMC_PCDRx, and PMC_PCSRx is the Peripheral Identifier defined at the product level. The bit number corresponds to the interrupt source number assigned to the peripheral.

31.9 Asynchronous Partial Wake-up

31.9.1 Description

The asynchronous partial wake-up wakes up a peripheral in a fully asynchronous way when activity is detected on the communication line. The asynchronous partial wake-up function automatically manages the peripheral clock. It reduces overall power consumption of the system by clocking peripherals only when needed.

Asynchronous partial wake-up can be enabled in Wait mode (SleepWalking), or in Active mode.

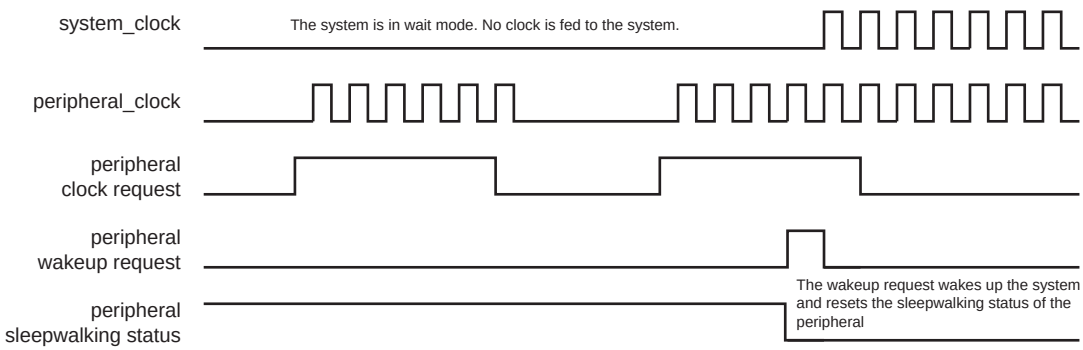
Only the following peripherals can be configured with asynchronous partial wake-up: UARTx and TWIHSx.

The peripheral selected for asynchronous partial wake-up must first be configured so that its clock is enabled. To do so, write a ‘1’ to the appropriate PIDx bit in PMC_PCER registers.

31.9.2 Asynchronous Partial Wake-up in Wait Mode (SleepWalking)

When the system is in Wait mode, all clocks of the system (except SLCK) are stopped. When an asynchronous clock request from a peripheral occurs, the PMC partially wakes up the system to feed the clock only to this peripheral. The rest of the system is not fed with the clock, thus optimizing power consumption. Finally, depending on user-configurable conditions, the peripheral either wakes up the whole system if these conditions are met or stops the peripheral clock until the next clock request. If a wake-up request occurs, SleepWalking is automatically disabled until the user instructs the PMC to enable SleepWalking. This is done by writing a ‘1’ to PIDx in the PMC SleepWalking Enable register (PMC_SLPWK_ER).

Figure 31-2. SleepWalking Waveforms



31.9.2.1 Configuration Procedure

Before configuring SleepWalking for a peripheral, check that the PIDx bit in PMC_PCSR is set. This ensures that the peripheral clock is enabled.

To enable SleepWalking for a peripheral, follow the steps below:

1. Check that the corresponding PIDx bit in the PMC SleepWalking Activity Status register (PMC_SLPWK_ASR) is set to ‘0’. This ensures that the peripheral has no activity in progress.
2. Enable SleepWalking for the peripheral by writing a ‘1’ to the corresponding PIDx bit in the PMC_SLPWK_ER.
3. Check that the corresponding PIDx bit in PMC_SLPWK_ASR is set to ‘0’. This ensures that no activity has started during the enable phase.
4. In the PMC_SLPWK_ASR, if the corresponding PIDx bit is set, SleepWalking must be immediately disabled by writing a ‘1’ to the PIDx bit in the PMC SleepWalking Disable register (PMC_SLPWK_DR). Wait for the

Downloaded from Arrow.com.

end of peripheral activity before reinitializing the procedure.
If the corresponding PIDx bit is set to ‘0’, then the peripheral clock is disabled and the system can now be placed in Wait mode.

Before entering Wait mode, check that the AIP bit in the PMC SleepWalking Activity In Progress register (PMC_SLPWK_AIPR) is cleared. This ensures that none of the peripherals is currently active.

Note: When SleepWalking for a peripheral is enabled and the core is running (system not in Wait mode), the peripheral must not be accessed before a wake-up of the peripheral is performed.

31.9.3 Asynchronous Partial Wake-Up in Active Mode

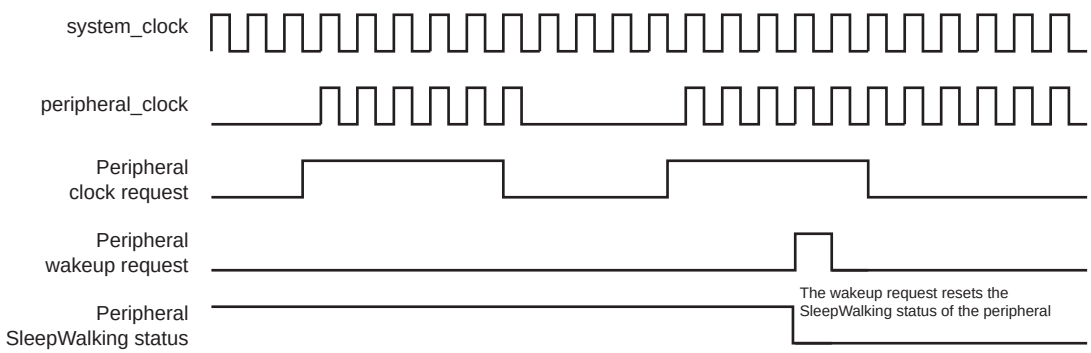
When the system is in Active mode, peripherals enabled for asynchronous partial wake-up have their respective clocks stopped until the peripherals request a clock. When a peripheral requests the clock, the PMC provides the clock without processor intervention.

The triggering of the peripheral clock request depends on conditions which can be configured for each peripheral. If these conditions are met, the peripheral asserts a request to the PMC. The PMC disables the Asynchronous Partial Wake-up mode of the peripheral and provides the clock to the peripheral until the user instructs the PMC to re-enable partial wake-up on the peripheral. This is done by setting PMC_SLPWK_ER.PIDx.

If the conditions are not met, the peripheral clears the clock request and the PMC stops the peripheral clock until the clock request is re-asserted by the peripheral.

Note: Configuring Asynchronous Partial Wake-up mode requires the same registers as SleepWalking mode.

Figure 31-3. Asynchronous Partial Wake-up in Active Mode



31.9.3.1 Configuration Procedure

Before configuring the asynchronous partial wake-up function of a peripheral, check that the PIDx bit in PMC_PCSR is set. This ensures that the peripheral clock is enabled.

To enable the asynchronous partial wake-up function of a peripheral, follow the steps below:

1. Check that the corresponding PIDx bit in the PMC SleepWalking Activity Status register (PMC_SLPWK_ASR) is set to ‘0’. This ensures that the peripheral has no activity in progress.
2. Enable the asynchronous partial wake-up function of the peripheral by writing a ‘1’ to the corresponding PIDx bit in the PMC_SLPWK_ER.
3. Check that the corresponding PIDx bit in PMC_SLPWK_ASR is set to ‘0’. This ensures that no activity has started during the enable phase.

If an activity has started during the enable phase, the asynchronous partial wake-up function must be immediately disabled by writing a ‘1’ to the PIDx bit in the PMC SleepWalking Disable register (PMC_SLPWK_DR). Wait for the end of peripheral activity before reinitializing the procedure.

Downloaded from Arrow.com.

31.10 Free-Running Processor Clock

The free-running processor clock (FCLK) used for sampling interrupts and clocking debug blocks ensures that interrupts can be sampled, and sleep events can be traced, while the processor is sleeping.

31.11 Programmable Clock Output Controller

The PMC controls three signals to be output on the external pins PCKx. Each signal can be independently programmed via the Programmable Clock registers (PMC_PCKx).

PCKx can be independently selected between the slow clock (SLCK), the main clock (MAINCK), the PLLA clock (PLLACK), UTMI PLL clock divided by 1 or 2 (UPLLCKDIV) and the master clock (MCK) by configuring PMC_PCKx.CSS. Each output signal can also be divided by a power of 2 between 1 and 64 by configuring PMC_PCKx.PRES.

Each output signal can be enabled and disabled by writing 1 in the corresponding PMC_SCER.PCKx and PMC_SCDR.PCKx, respectively. Status of the active programmable output clocks are given in PMC_SCSR.PCKx

The status flag PMC_SR.PCKRDYx indicates that the programmable clock is actually what has been programmed in registers PMC_PCKx.

As the Programmable Clock Controller does not manage with glitch prevention when switching clocks, it is strongly recommended to disable the programmable clock before any configuration change and to re-enable it after the change is performed.

31.12 Core and Bus Independent Clocks for Peripherals

Table 31-1 lists the peripherals that can operate while the core, bus and peripheral clock frequencies are modified, thus providing communications at a rate which is independent for the core/bus/peripheral clock. This mode of operation is possible by using the internally generated independent clock sources.

Internal clocks can be independently selected between the slow clock (SLCK), the main clock (MAINCK), any available PLL clock, and the master clock (MCK) by configuring PMC_PCKx.CSS. The independent clock sources can be also divided by configuring PMC_PCKx.PRES.

Each internal clock signal (PCKx) can be enabled and disabled by writing a '1' to the corresponding PMC_SCER.PCKx and PMC_SCDR.PCKx, respectively. The status of the internal clocks are given in PMC_SCSR.PCKx.

The status flag PMC_SR.PCKRDYx indicates that the programmable internal clock has been programmed in the programmable clock registers.

The independent clock source must also be selected in each peripheral in Table 31-1 to operate communications, timings, etc without influencing the frequency of the core/bus/peripherals (except frequency limitations listed in each peripheral).

Table 31-1. Clock Assignment

Clock Name	Peripheral
PCK3	ETM
PCK4	UART/USART
PCK5	CAN
PCK6	TC

31.13 Fast Startup

At exit from Wait mode, the device allows the processor to restart in less than 10 microseconds only if the C-code function that manages the Wait mode entry and exit is linked to and executed from on-chip SRAM.

Downloaded from Arrow.com.

The fast startup time cannot be achieved if the first instruction after an exit is located in the embedded Flash.

If fast startup is not required, or if the first instruction after exit from Wait mode is located in embedded Flash, see Section 31.14 ”Startup from Embedded Flash”.

Prior to instructing the device to enter Wait mode:

1. Select the 4/8/12 MHz RC oscillator as the master clock source by configuring PMC_MCKR.CSS to ‘1’.
2. Disable the PLL if enabled.
3. Wait for two SLCK clock cycles.
4. Clear the internal wake-up sources.
5. Verify that none of the enabled external wake-up inputs (WKUP) hold an active polarity.

The system enters Wait mode either by setting the WAITMODE bit in CKGR_MOR, or by executing the WaitForEvent (WFE) instruction of the processor while the LPM bit is at ‘1’ in PMC_FSMR. Immediately after setting the WAITMODE bit or using the WFE instruction, wait for the MCKRDY bit to be set in PMC_SR.

A fast startup is enabled upon the detection of a programmed level on one of the 14 wake-up inputs (WKUP) or upon an active alarm from the RTC, RTT and USB Controller. The polarity of the 14 wake-up inputs is programmable by writing the PMC Fast Startup Polarity register (PMC_FSPR).

WARNING: The duration of the WKUPx pins active level must be greater than four main clock cycles.

The fast startup circuitry, as shown in Figure 31-4, is fully asynchronous and provides a fast startup signal to the PMC. As soon as the fast startup signal is asserted, the embedded 4/8/12 MHz RC oscillator restarts automatically.

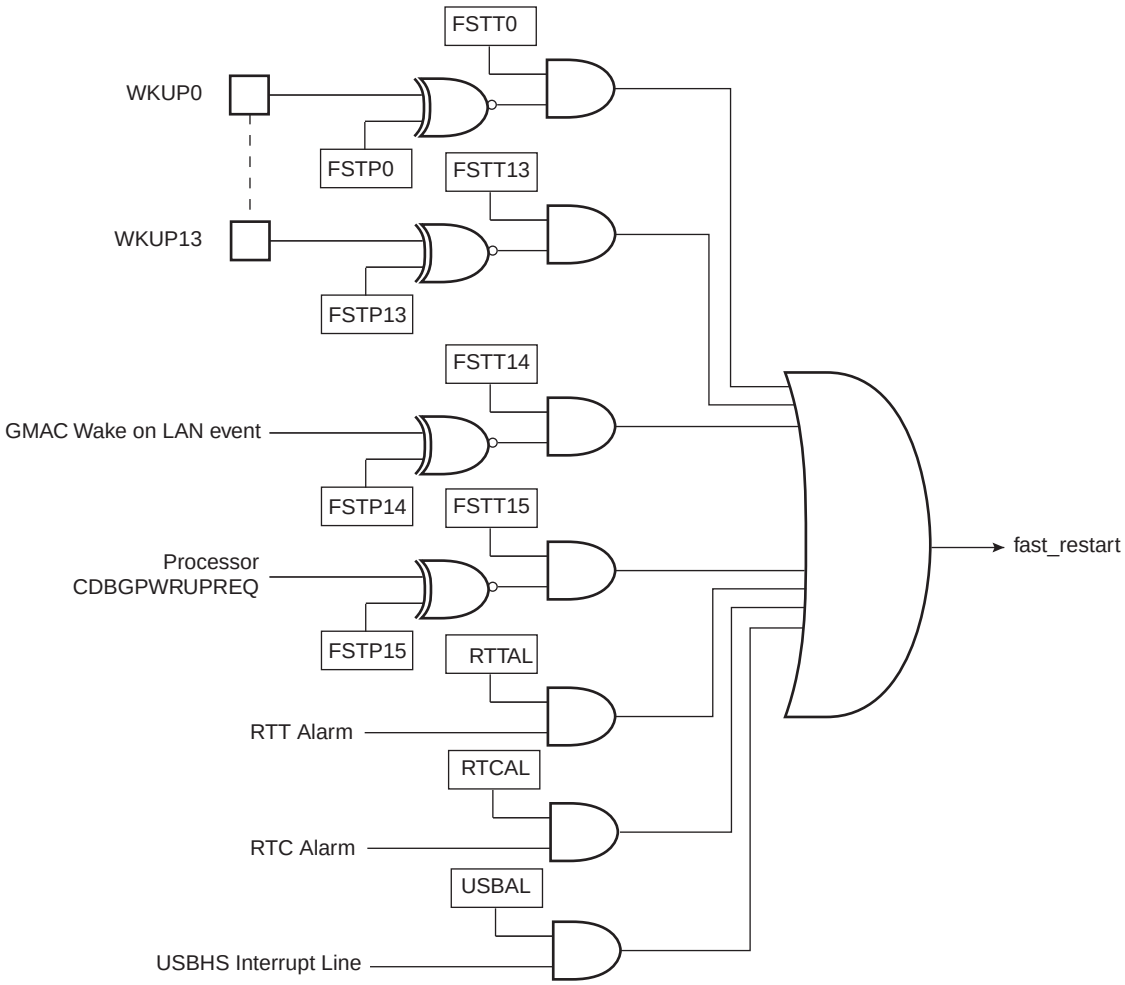
When entering Wait mode, the embedded Flash can be placed in one of the low-power modes (Deep-power-down or Standby mode) with PMC_FSMR.FLPM. FLPM can be configured at any time and its value will be applied to the next Wait mode period.

The power consumption reduction is optimal when PMC_FSMR.FLPM is configured to ‘1’ (Deep-power-down mode). If the field is configured to ‘0’ (Standby mode), the power consumption is slightly higher than in Deep-power-down mode.

When PMC_FSMR.FLPM is configured to ‘2’, the Wait mode Flash power consumption is equivalent to that of the Active mode when there is no read access on the Flash.

Downloaded from Arrow.com.

Figure 31-4. Fast Startup Circuitry



Each wake-up input pin and alarm can be enabled to generate a fast startup event by setting the corresponding bit in PMC_FSMR.

The user interface does not provide any status for fast startup. The status can be read in the PIO Controller and the status registers of the RTC, RTT and USB Controller.

31.14 Startup from Embedded Flash

The inherent start-up time of the embedded Flash cannot provide a fast startup of the system.

If system fast start-up time is not required, the first instruction after a Wait mode exit can be located in the embedded Flash. Under these conditions, prior to entering Wait mode, the Flash controller must be programmed to perform access in 0 wait-state (refer to Section 21. ”Enhanced Embedded Flash Controller (EEFC)”).

The procedure and conditions to enter Wait mode and the circuitry to exit Wait mode are strictly the same as fast startup (see Section 31.13 ”Fast Startup”).

31.15 Main Clock Failure Detection

The clock failure detector monitors the 3 to 20 MHz crystal oscillator or ceramic resonator-based oscillator to identify a failure of this oscillator when selected as main clock.

Downloaded from Arrow.com.

The clock failure detector can be enabled or disabled by configuring CKGR_MOR.CFDEN. After a VDDCORE reset, the detector is disabled. However, if the oscillator is disabled (MOSCXTEN = 0), the detector is also disabled.

To initialize the clock failure detector, follow the sequence below:

1. The 4/8/12 MHz RC oscillator must be selected as the source of MAINCK.
2. MCK must select MAINCK.
3. Enable the clock failure detector by writing a '1' to CFDEN.
4. PMC_SR must be read two slow clock cycles after enabling the clock failure detector. The value read is meaningless.

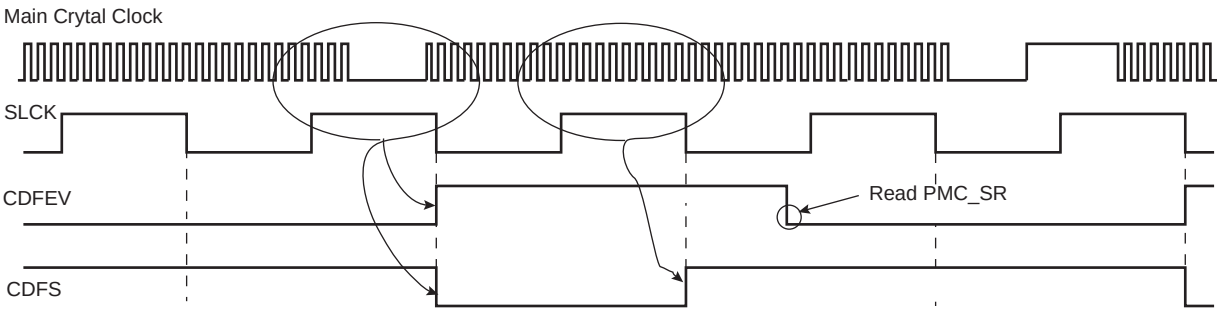
The clock failure detector is now initialized and MCK can select another clock source by configuring PMC_MCKR.CSS.

A failure is detected by means of a counter incrementing on the main clock and detection logic is triggered by the 32 kHz (typical) RC oscillator which is automatically enabled when CFDEN=1.

The counter is cleared when the 32 kHz (typical) RC oscillator clock signal is low and enabled when the signal is high. Thus, the failure detection time is one RC oscillator period. If, during the high level period of the 32 kHz (typical) RC oscillator clock signal, less than eight 3 to 20 MHz crystal oscillator clock periods have been counted, then a failure is reported.

If a failure of the main clock is detected, PMC_SR.CFDEV indicates a failure event and generates an interrupt if the corresponding interrupt source is enabled. The interrupt remains active until a read occurs in PMC_SR. The status of the clock failure detection can be read at any time from PMC_SR.CFDS.

Figure 31-5. Clock Failure Detection Example



Note: ratio of clock periods is for illustration purposes only

If the 3 to 20 MHz crystal oscillator or ceramic resonator-based oscillator is selected as the source clock of MAINCK (CKGR_MOR.MOSCSEL = 1), and if MCK source is PLLACK or UPLLCKDIV (CSS = 2 or 3), a clock failure detection automatically forces the MAINCK to be the source clock for the master clock MCK. Then, regardless of the PMC configuration, a clock failure detection automatically forces the 4/8/12 MHz RC oscillator to be the source clock for MAINCK. If the fast RC oscillator is disabled when a clock failure detection occurs, it is automatically re-enabled by the clock failure detection mechanism.

It takes two 32 kHz (typical) RC oscillator clock cycles to detect and switch from the 3 to 20 MHz crystal oscillator, to the 4/8/12 MHz RC oscillator if the source master clock (MCK) is main clock (MAINCK), or three 32 kHz (typical) RC oscillator clock cycles if the source of MCK is PLLACK or UPLLCKDIV.

A clock failure detection activates a fault output that is connected to the Pulse Width Modulator (PWM) Controller. With this connection, the PWM controller is able to force its outputs and to protect the driven device, if a clock failure is detected.

The user can know the status of the clock failure detector at any time by reading the FOS bit in PMC_SR.

Downloaded from Arrow.com.

This fault output remains active until the defect is detected and until it is cleared by the bit FOCLR in the PMC Fault Output Clear register (PMC_FOCR).

31.16 32.768 kHz Crystal Oscillator Frequency Monitor

The frequency of the 32.768 kHz crystal oscillator can be monitored by means of logic driven by the 4/8/12 MHz RC oscillator known as a reliable clock source. This function is enabled by configuring the XT32KFME bit of CKGR_MOR. The SEL4/SEL8/SEL12 bits of PMC_OCR must be cleared.

An error flag (XT32KERR in PMC_SR) is asserted when the 32.768 kHz crystal oscillator frequency is out of the ±10% nominal frequency value (i.e. 32.768 kHz). The error flag can be cleared only if the slow clock frequency monitoring is disabled.

When the 4/8/12 MHz RC oscillator frequency is 4 MHz, the accuracy of the measurement is ±40% as this frequency is not trimmed during production. Therefore, ±10% accuracy is obtained only if the RC oscillator frequency is configured for 8 or 12 MHz.

The monitored clock frequency is declared invalid if at least 4 consecutive clock period measurement results are over the nominal period ±10%.

Due to the possible frequency variation of the 4/8/12 MHz RC oscillator acting as reference clock for the monitor logic, any 32.768 kHz crystal frequency deviation over ±10% of the nominal frequency is systematically reported as an error by means of XT32KERR in PMC_SR. Between -1% and -10% and +1% and +10%, the error is not systematically reported.

Thus only a crystal running at 32.768 kHz frequency ensures that the error flag will not be asserted. The permitted drift of the crystal is 10000ppm (1%), which allows any standard crystal to be used.

If the 4/8/12 MHz RC oscillator frequency needs to be changed while the slow clock frequency monitor is operating, the monitoring must be stopped prior to change the 4/8/12 MHz RC oscillator frequency. Then it can be re-enabled as soon as MOSCRCS is set in PMC_SR.

The error flag can be defined as an interrupt source of the PMC by setting the XT32KERR bit of PMC_IER.

31.17 Programming Sequence

1. If the 3 to 20 MHz crystal oscillator is not required, the PLL and divider can be directly configured (Step 6.) else this oscillator must be started (Step 2.).
2. Enable the 3 to 20 MHz crystal oscillator by setting CKGR_MOR.MOSCXTEN. The user can define a start-up time. This can be achieved by writing a value in the CKGR_MOR.MOSCXTST. Once this register has been correctly configured, the user must wait for PMC_SR.MOSCXTS to be set. This can be done either by polling PMC_SR.MOSCXTS, or by waiting for the interrupt line to be raised if the associated interrupt source (MOSCXTS) has been enabled in PMC_IER.
3. Switch the main clock to the 3 to 20 MHz crystal oscillator by setting CKGR_MOR.MOSCSEL.
4. Wait for PMC_SR.MOSCSELS to be set to ensure the switchover is complete.
5. Check the main clock frequency:

   This frequency can be measured via CKGR_MCFR.

   Read CKGR_MCFR until the MAINFRDY field is set, after which the user can read CKGR_MCFR.MAINF by performing an additional read. This provides the number of main clock cycles that have been counted during a period of 16 slow clock cycles.

   If MAINF = 0, switch the main clock to the 4/8/12 RC Oscillator by clearing CKGR_MOR.MOSCSEL. If MAINF ≠ 0, proceed to Step 6.
6. Set PLLx and Divider (if not required, proceed to Step 7.):

   In the names PLLx, DIVx, MULx, LOCKx, PLLxCOUNT, and CKGR_PLLxR, ‘x’ represents A.

Downloaded from Arrow.com.

All parameters needed to configure PLLx and the divider are located in CKGR_PLLxR.

CKGR_PLLxR.DIVx is used to control the divider itself. This parameter can be programmed between 0 and 127. Divider output is divider input divided by DIVx parameter. By default, DIVx field is cleared which means that the divider and PLLx are turned off.

CKGR_PLLxR.MULx is the PLLx multiplier factor. This parameter can be programmed between 0 and 62. If MULx is cleared, PLLx will be turned off, otherwise the PLLx output frequency is PLLx input frequency multiplied by (MULx + 1).

CKGR_PLLxR.PLLxCOUNT specifies the number of slow clock cycles before PMC_SR.LOCKx is set after CKGR_PLLxR has been written.

Once CKGR_PLLxR has been written, the user must wait for PMC_SR.LOCKx to be set. This can be done either by polling PMC_SR.LOCKx or by waiting for the interrupt line to be raised if the associated interrupt source (LOCKx) has been enabled in PMC_IER. All fields in CKGR_PLLxR can be programmed in a single write operation. If MULx or DIVx is modified, the LOCKx bit goes low to indicate that PLLx is not yet ready. When PLLx is locked, LOCKx is set again. The user must wait for the LOCKx bit to be set before using the PLLx output clock.

7. Select the master clock and processor clock:

   The master clock and the processor clock are configurable via PMC_MCKR.

   PMC_MCKR.CSS is used to select the clock source of the master clock and processor clock dividers. By default, the selected clock source is the main clock.

   PMC_MCKR.PRES is used to define the processor clock and master clock prescaler. The user can choose between different values (1, 2, 3, 4, 8, 16, 32, 64). Prescaler output is the selected clock source frequency divided by the PRES value.

   PMC_MCKR.MDIV is used to define the master clock divider. It is possible to choose between different values (0, 1, 2, 3). The master clock output is the processor clock frequency divided by 1, 2, 3 or 4, depending on the value programmed in MDIV.

   By default, MDIV is cleared, which indicates that the processor clock is equal to the master clock.

   Once the PMC_MCKR has been written, the user must wait for PMC_SR.MCKRDY to be set. This can be done either by polling PMC_SR.MCKRDY or by waiting for the interrupt line to be raised if the associated interrupt source (MCKRDY) has been enabled in PMC_IER. PMC_MCKR must not be programmed in a single write operation. The programming sequence for PMC_MCKR is as follows:

   If a new value for PMC_MCKR.CSS corresponds to PLL clock:

   a. Program the PMC_MCKR.PRES.
   b. Wait for PMC_SR.MCKRDY to be set.
   c. Program the PMC_MCKR.MDIV.
   d. Wait for PMC_SR.MCKRDY to be set.
   e. Program the PMC_MCKR.CSS.
   f. Wait for PMC_SR.MCKRDY to be set.

   If a new value for PMC_MCKR.CSS corresponds to MAINCK or SLCK:

   a. Program PMC_MCKR.CSS.
   b. Wait for PMC_SR.MCKRDY to be set.
   c. Program PMC_MCKR.PRES.
   d. Wait for PMC_SR.MCKRDY to be set.

   If CSS, MDIV or PRES are modified at any stage, the MCKRDY bit goes low to indicate that the master clock and the processor clock are not yet ready. The user must wait for MCKRDY bit to be set again before using the master and processor clocks.

Downloaded from Arrow.com.

Note: IF PLLx clock was selected as the master clock and the user decides to modify it by writing in CKGR_PLLxR, the MCKRDY flag will go low while PLLx is unlocked. Once PLLx is locked again, LOCKx goes high and MCKRDY is set. While PLLx is unlocked, the master clock selection is automatically changed to slow clock for PLLA. For further information, see Section 31.18.2 "Clock Switching Waveforms".

The master clock is the main clock divided by 2.

8. Select the programmable clocks:

   Programmable clocks are controlled via registers PMC_SCER, PMC_SCDR and PMC_SCSR.

   Programmable clocks can be enabled and/or disabled via PMC_SCER and PMC_SCDR. Three programmable clocks can be used. PMC_SCSR indicates which programmable clock is enabled. By default all programmable clocks are disabled.

   PMC_PCKx registers are used to configure programmable clocks.

   PMC_PCKx.CSS is used to select the programmable clock divider source. Several clock options are available: main clock, slow clock, master clock, PLLACK and UPLLCKDIV. The slow clock is the default clock source.

   PMC_PCKx.PRES is used to control the programmable clock prescaler. It is possible to choose between different values (1, 2, 4, 8, 16, 32, 64). Programmable clock output is prescaler input divided by PRES. By default, the PRES value is cleared which means that PCKx is equal to slow clock.

   Once PMC_PCKx has been configured, the corresponding programmable clock must be enabled and the user must wait for PMC_SR.PCKRDYx to be set. This can be done either by polling PMC_SR.PCKRDYx or by waiting for the interrupt line to be raised if the associated interrupt source (PCKRDYx) has been enabled in PMC_IER. All parameters in PMC_PCKx can be programmed in a single write operation.

   If the PMC_PCKx.CSS and PMC_PCKx.PRES parameters are to be modified, the corresponding programmable clock must be disabled first. The parameters can then be modified. Once this has been done, the user must re-enable the programmable clock and wait for the PCKRDYx bit to be set.

9. Enable the peripheral clocks

   Once all of the previous steps have been completed, the peripheral clocks can be enabled and/or disabled via registers PMC_PCERx and PMC_PCDRx.

31.18 Clock Switching Details

31.18.1 Master Clock Switching Timings

Table 31-2 and Table 31-3 give the worst case timings required for the master clock to switch from one selected clock to another one. This is in the event that the prescaler is de-activated. When the prescaler is activated, an additional time of 64 clock cycles of the newly selected clock has to be added.

Table 31-2. Clock Switching Timings (Worst Case)

From To	MAINCK	SLCK	PLL Clock
Main Clock	–	4 x SLCK + 2.5 x MAINCK	3 x PLL Clock + 4 x SLCK + 1 x MAINCK
SLCK	0.5 x MAINCK + 4.5 x SLCK	–	3 x PLL Clock + 5 x SLCK
PLL Clock	0.5 x MAINCK + 4 x SLCK + PLLCOUNT x SLCK + 2.5 x PLLx Clock	2.5 x PLL Clock + 5 x SLCK + PLLCOUNT x SLCK	2.5 x PLL Clock + 4 x SLCK + PLLCOUNT x SLCK

Downloaded from Arrow.com.

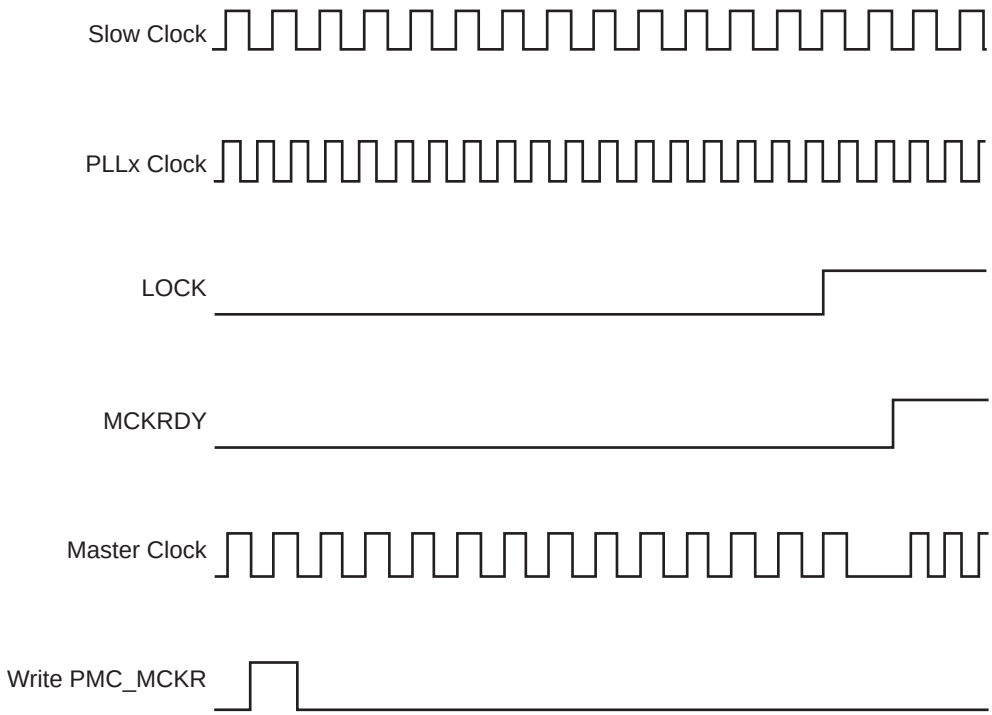
Notes: 1. PLL designates either the PLLA or the UPLL clock.
2. PLLCOUNT designates either PLLACOUNT or UPLLCOUNT.

Table 31-3. Clock Switching Timings between Two PLLs (Worst Case)

From / To	PLLA Clock	UPLL Clock
PLLA Clock	2.5 x PLLA Clock + 4 x SLCK + PLLACOUNT x SLCK	3 x PLLA Clock + 4 x SLCK + 1.5 x PLLA Clock
UPLL Clock	3 x UPLL Clock + 4 x SLCK + 1.5 x UPLL Clock	2.5 x UPLL Clock + 4 x SLCK

31.18.2 Clock Switching Waveforms

Figure 31-6. Switch Master Clock from Slow Clock to PLLx Clock



Downloaded from Arrow.com.

Figure 31-7. Switch Master Clock from Main Clock to Slow Clock

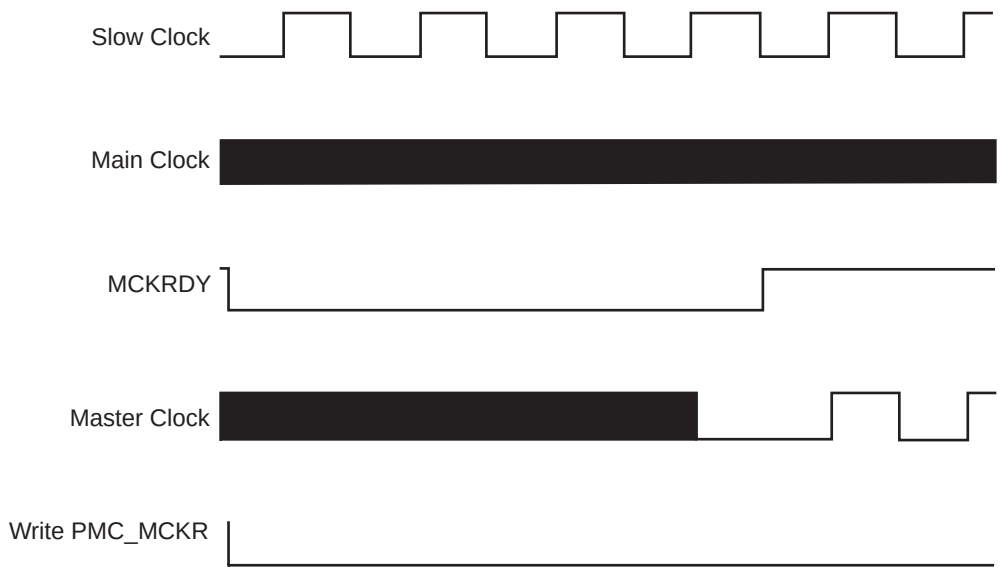
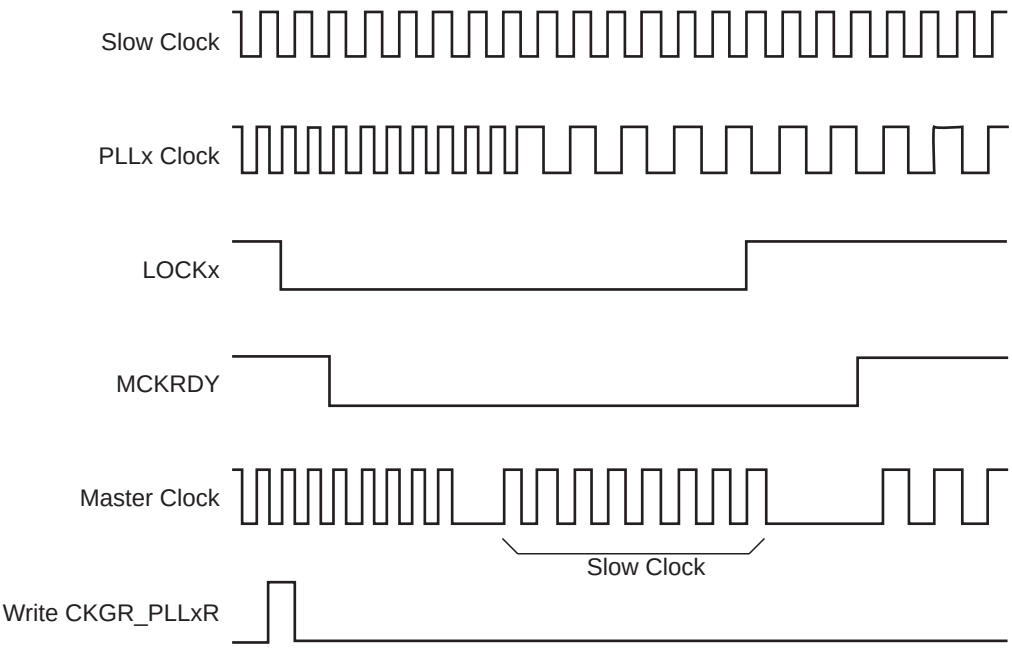
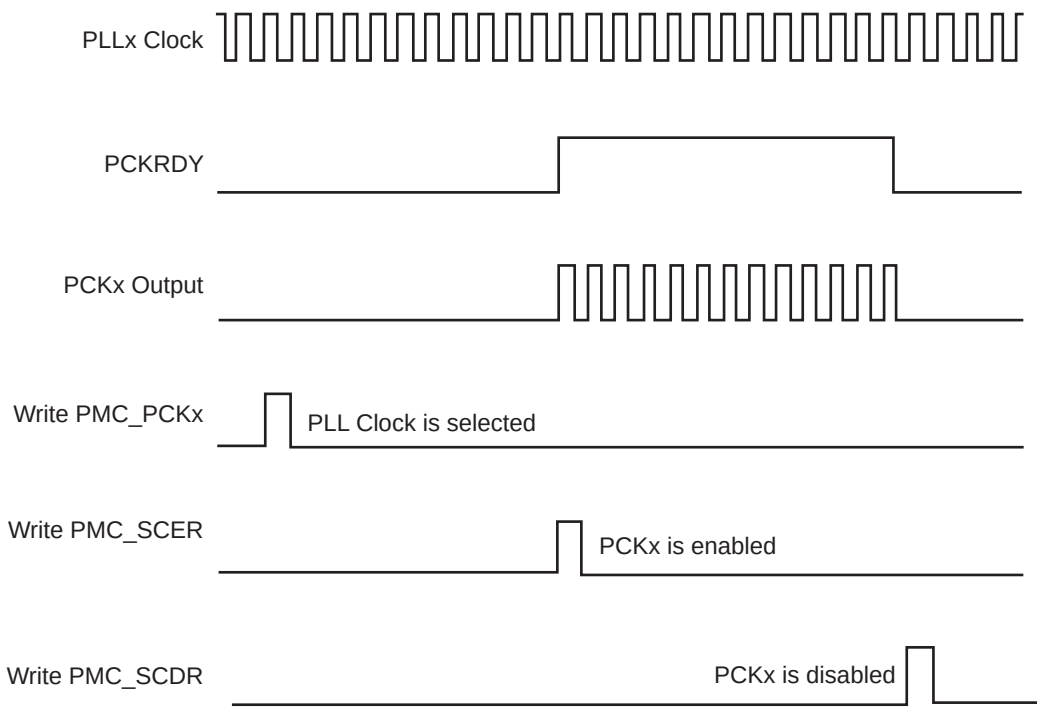


Figure 31-8. Change PLLx Programming



Downloaded from Arrow.com.

Figure 31-9. Programmable Clock Output Programming



31.19 Register Write Protection

To prevent any single software error from corrupting PMC behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the PMC Write Protection Mode Register (PMC_WPMR).

If a write access to a write-protected register is detected, the WPVS flag in the PMC Write Protection Status Register (PMC_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading the PMC_WPSR.

The following registers are write-protected when the WPEN bit is set in PMC_WPMR:

- PMC System Clock Disable Register
- PMC Peripheral Clock Enable Register 0
- PMC Peripheral Clock Disable Register 0
- PMC Clock Generator Main Oscillator Register
- PMC Clock Generator Main Clock Frequency Register
- PMC Clock Generator PLLA Register
- PMC UTMI Clock Configuration Register
- PMC Master Clock Register
- PMC USB Clock Register
- PMC Programmable Clock Register
- PMC Fast Startup Mode Register
- PMC Fast Startup Polarity Register
- PMC Peripheral Clock Enable Register 1

Downloaded from Arrow.com.

- PMC Peripheral Clock Disable Register 1
- PMC Oscillator Calibration Register
- PMC SleepWalking Enable Register 0
- PMC SleepWalking Disable Register 0
- PLL Maximum Multiplier Value Register
- PMC SleepWalking Enable Register 1
- PMC SleepWalking Disable Register 1

Downloaded from Arrow.com.

31.20 Power Management Controller (PMC) User Interface

Table 31-4. Register Mapping

Offset	Register	Name	Access	Reset
0x0000	System Clock Enable Register	PMC_SCER	Write-only	–
0x0004	System Clock Disable Register	PMC_SCDR	Write-only	–
0x0008	System Clock Status Register	PMC_SCSR	Read-only	0x0000_0001
0x000C	Reserved	–	–	–
0x0010	Peripheral Clock Enable Register 0	PMC_PCER0	Write-only	–
0x0014	Peripheral Clock Disable Register 0	PMC_PCDR0	Write-only	–
0x0018	Peripheral Clock Status Register 0	PMC_PCSR0	Read-only	0x0000_0000
0x001C	UTMI Clock Register	CKGR_UCKR	Read/Write	0x1020_0800
0x0020	Main Oscillator Register	CKGR_MOR	Read/Write	0x0000_0008
0x0024	Main Clock Frequency Register	CKGR_MCFR	Read/Write	0x0000_0000
0x0028	PLLA Register	CKGR_PLLAR	Read/Write	0x0000_3F00
0x002C	Reserved	–	–	–
0x0030	Master Clock Register	PMC_MCKR	Read/Write	0x0000_0001
0x0034	Reserved	–	–	–
0x0038	USB Clock Register	PMC_USB	Read/Write	0x0000_0000
0x003C	Reserved	–	–	–
0x0040+chid*0x04	Programmable Clock Register	PMC_PCK	Read/Write	0x0000_0000
0x005C	Reserved	–	–	–
0x0060	Interrupt Enable Register	PMC_IER	Write-only	–
0x0064	Interrupt Disable Register	PMC_IDR	Write-only	–
0x0068	Status Register	PMC_SR	Read-only	0x0003_0008
0x006C	Interrupt Mask Register	PMC_IMR	Read-only	0x0000_0000
0x0070	Fast Startup Mode Register	PMC_FSMR	Read/Write	0x0000_0000
0x0074	Fast Startup Polarity Register	PMC_FSPR	Read/Write	0x0000_0000
0x0078	Fault Output Clear Register	PMC_FOCR	Write-only	–
0x007C–0x00E0	Reserved	–	–	–
0x00E4	Write Protection Mode Register	PMC_WPMR	Read/Write	0x0
0x00E8	Write Protection Status Register	PMC_WPSR	Read-only	0x0
0x00EC–0x00FC	Reserved	–	–	–
0x0100	Peripheral Clock Enable Register 1	PMC_PCER1	Write-only	–
0x0104	Peripheral Clock Disable Register 1	PMC_PCDR1	Write-only	–
0x0108	Peripheral Clock Status Register 1	PMC_PCSR1	Read-only	0x0000_0000
0x010C	Peripheral Control Register	PMC_PCR	Read/Write	0x0000_0000
0x0110	Oscillator Calibration Register	PMC_OCR	Read/Write	(2)
0x0114	SleepWalking Enable Register 0	PMC_SLPWK_ER0	Write-only	–

Downloaded from Arrow.com.

Table 31-4. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x0118	SleepWalking Disable Register 0	PMC_SLPWK_DR0	Write-only	–
0x011C	SleepWalking Status Register 0	PMC_SLPWK_SR0	Read-only	0x00000000
0x0120	SleepWalking Activity Status Register 0	PMC_SLPWK_ASR0	Read-only	0x00000000
0x0130	PLL Maximum Multiplier Value Register	PMC_PMMR	Read/Write	0x0000_07FF
0x0134	SleepWalking Enable Register 1	PMC_SLPWK_ER1	Write-only	–
0x0138	SleepWalking Disable Register 1	PMC_SLPWK_DR1	Write-only	–
0x013C	SleepWalking Status Register 1	PMC_SLPWK_SR1	Read-only	0x00000000
0x0140	SleepWalking Activity Status Register 1	PMC_SLPWK_ASR1	Read-only	0x00000000
0x0144	SleepWalking Activity In Progress Register	PMC_SLPWK_AIPR	Read-only	–

Notes: 1. If an offset is not listed in Table 31-4 it must be considered as “reserved”.
2. The reset value depends on factory settings.

Downloaded from Arrow.com.

31.20.1 PMC System Clock Enable Register

Name: PMC_SCER

Address: 0x400E0600

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	PCK6	PCK5	PCK4	PCK3	PCK2	PCK1	PCK0
7	6	5	4	3	2	1	0
–	–	USBCLK	–	–	–	–	–

This register can only be written if the WPEN bit is cleared in PMC Write Protection Mode Register.

- **USBCLK: Enable USB FS Clock**

0: No effect.

1: Enables USB FS clock.

- **PCKx: Programmable Clock x Output Enable**

0: No effect.

1: Enables the corresponding Programmable Clock output.

Downloaded from Arrow.com.

31.20.2 PMC System Clock Disable Register

Name: PMC_SCDR

Address: 0x400E0604

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	PCK6	PCK5	PCK4	PCK3	PCK2	PCK1	PCK0

7	6	5	4	3	2	1	0
–	–	USBCLK	–	–	–	–	–

This register can only be written if the WPEN bit is cleared in PMC Write Protection Mode Register.

- **USBCLK: Disable USB FS Clock**

0: No effect.

1: Disables USB FS clock.

- **PCKx: Programmable Clock x Output Disable**

0: No effect.

1: Disables the corresponding Programmable Clock output.

Downloaded from Arrow.com.

31.20.3 PMC System Clock Status Register

Name: PMC_SCSR

Address: 0x400E0608

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	PCK6	PCK5	PCK4	PCK3	PCK2	PCK1	PCK0

7	6	5	4	3	2	1	0
–	–	USBCLK	–	–	–	–	HCLKS

- **HCLKS: Processor Clock Status**

0: The processor clock is disabled.

1: The processor clock is enabled.

- **USBCLK: USB FS Clock Status**

0: The USB FS clock is disabled.

1: The USB FS clock is enabled.

- **PCKx: Programmable Clock x Output Status**

0: The corresponding Programmable Clock output is disabled.

1: The corresponding Programmable Clock output is enabled.

Downloaded from Arrow.com.

31.20.4 PMC Peripheral Clock Enable Register 0

Name: PMC_PCER0

Address: 0x400E0610

Access: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	PID8
7	6	5	4	3	2	1	0
PID7	–	–	–	–	–	–	–

This register can only be written if the WPEN bit is cleared in PMC Write Protection Mode Register.

- **PIDx: Peripheral Clock x Enable**

0: No effect.

1: Enables the corresponding peripheral clock.

Note: PIDx refers to identifiers defined in the section “Peripheral Identifiers”. Other peripherals can be enabled in PMC_PCER1 (Section 31.20.23 ”PMC Peripheral Clock Enable Register 1”).

Note: Programming the control bits of the Peripheral ID that are not implemented has no effect on the behavior of the PMC.

Downloaded from Arrow.com.

31.20.5 PMC Peripheral Clock Disable Register 0

Name: PMC_PCDR0

Address: 0x400E0614

Access: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	PID8
7	6	5	4	3	2	1	0
PID7	–	–	–	–	–	–	–

This register can only be written if the WPEN bit is cleared in PMC Write Protection Mode Register.

- **PIDx: Peripheral Clock x Disable**

0: No effect.

1: Disables the corresponding peripheral clock.

Note: PIDx refers to identifiers defined in the section “Peripheral Identifiers”. Other peripherals can be disabled in PMC_PCDR1 (Section 31.20.24 ”PMC Peripheral Clock Disable Register 1”).

Downloaded from Arrow.com.

31.20.6 PMC Peripheral Clock Status Register 0

Name: PMC_PCSR0

Address: 0x400E0618

Access: Read-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	PID8
7	6	5	4	3	2	1	0
PID7	–	–	–	–	–	–	–

- **PIDx: Peripheral Clock x Status**

0: The corresponding peripheral clock is disabled.

1: The corresponding peripheral clock is enabled.

Note: PIDx refers to identifiers defined in the section “Peripheral Identifiers”. Other peripherals status can be read in PMC_PCSR1 (Section 31.20.25 ”PMC Peripheral Clock Status Register 1”).

Downloaded from Arrow.com.

31.20.7 PMC UTMI Clock Configuration Register

Name: CKGR_UCKR

Address: 0x400E061C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
UPLLCOUNT				–	–	–	UPLLEN

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

This register can only be written if the WPEN bit is cleared in the PMC Write Protection Mode Register.

- **UPLLEN: UTMI PLL Enable**

0: The UTMI PLL is disabled.

1: The UTMI PLL is enabled.

When UPLLEN is set, the LOCKU flag is set once the UTMI PLL start-up time is achieved.

- **UPLLCOUNT: UTMI PLL Start-up Time**

Specifies the number of Slow Clock cycles multiplied by 8 for the UTMI PLL start-up time.

Downloaded from Arrow.com.

31.20.8 PMC Clock Generator Main Oscillator Register

Name: CKGR_MOR

Address: 0x400E0620

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	XT32KFME	CFDEN	MOSCSEL

23	22	21	20	19	18	17	16
KEY							

15	14	13	12	11	10	9	8
MOSCXTST							

7	6	5	4	3	2	1	0
–	MOSCRCF			MOSCRCEN	WAITMODE	MOSCXTBY	MOSCXTEN

This register can only be written if the WPEN bit is cleared in PMC Write Protection Mode Register.

- **MOSCXTEN: 3 to 20 MHz Crystal Oscillator Enable**

A crystal must be connected between XIN and XOUT.

0: The 3 to 20 MHz crystal oscillator is disabled.

1: The 3 to 20 MHz crystal oscillator is enabled. MOSCXTBY must be cleared.

When MOSCXTEN is set, the MOSCXTS flag is set once the Main Crystal Oscillator start-up time is achieved.

- **MOSCXTBY: 3 to 20 MHz Crystal Oscillator Bypass**

0: No effect.

1: The 3 to 20 MHz crystal oscillator is bypassed. MOSCXTEN must be cleared. An external clock must be connected on XIN.

When MOSCXTBY is set, the MOSCXTS flag in PMC_SR is automatically set.

Clearing MOSCXTEN and MOSCXTBY bits resets the MOSCXTS flag.

Note: When the crystal oscillator bypass is disabled (MOSCXTBY=0), the MOSCXTS flag must be read at 0 in PMC_SR before enabling the crystal oscillator (MOSCXTEN=1).

- **WAITMODE: Wait Mode Command (Write-only)**

0: No effect.

1: Puts the device in Wait mode.

- **MOSCRCEN: 4/8/12 MHz On-Chip RC Oscillator Enable**

0: The 4/8/12 MHz on-chip RC oscillator is disabled.

1: The 4/8/12 MHz on-chip RC oscillator is enabled.

When MOSCRCEN is set, the MOSCRCS flag is set once the on-chip RC oscillator start-up time is achieved.

Downloaded from Arrow.com.

- **MOSCRCF: 4/8/12 MHz RC Oscillator Frequency Selection**

At startup, the RC oscillator frequency is 4 MHz.

Value	Name	Description
0	4_MHz	The RC oscillator frequency is at 4 MHz (default)
1	8_MHz	The RC oscillator frequency is at 8 MHz
2	12_MHz	The RC oscillator frequency is at 12 MHz

Note: MOSCRCF must be changed only if MOSCRCS is set in the PMC_SR register. Therefore MOSCRCF and MOSCRCEN cannot be changed at the same time.

- **MOSCXTST: 3 to 20 MHz Crystal Oscillator Start-up Time**

Specifies the number of slow clock cycles multiplied by 8 for the main crystal oscillator start-up time.

- **KEY: Write Access Password**

Value	Name	Description
0x37	PASSWD	Writing any other value in this field aborts the write operation. Always reads as 0.

- **MOSCSEL: Main Clock Oscillator Selection**

0: The 4/8/12 MHz RC oscillator is selected.

1: The 3 to 20 MHz crystal oscillator is selected.

- **CFDEN: Clock Failure Detector Enable**

0: The clock failure detector is disabled.

1: The clock failure detector is enabled.

Note: 1. The 32 kHz (typical) RC oscillator must be enabled when CFDEN is enabled.

- **XT32KFME: 32.768 kHz Crystal Oscillator Frequency Monitoring Enable**

0: The 32.768 kHz crystal oscillator frequency monitoring is disabled.

1: The 32.768 kHz crystal oscillator frequency monitoring is enabled.

Downloaded from Arrow.com.

31.20.9 PMC Clock Generator Main Clock Frequency Register

Name: CKGR_MCFR

Address: 0x400E0624

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	CCSS

23	22	21	20	19	18	17	16
–	–	–	RCMEAS	–	–	–	MAINFRDY

15	14	13	12	11	10	9	8
MAINF							

7	6	5	4	3	2	1	0
MAINF							

This register can only be written if the WPEN bit is cleared in PMC Write Protection Mode Register.

- **MAINF: Main Clock Frequency**

Gives the number of cycles of the clock selected by the bit CCSS within 16 slow clock periods. To calculate the frequency of the measured clock:

$f_{SELCLK} = (MAINF \times f_{SLCK})/16$

where frequency is in MHz.

- **MAINFRDY: Main Clock Frequency Measure Ready**

0: MAINF value is not valid or the measured oscillator is disabled or a measure has just been started by means of RCMEAS.

1: The measured oscillator has been enabled previously and MAINF value is available.

Note: To ensure that a correct value is read on the MAINF field, the MAINFRDY flag must be read at 1 then another read access must be performed on the register to get a stable value on the MAINF field.

- **RCMEAS: RC Oscillator Frequency Measure (write-only)**

0: No effect.

1: Restarts measuring of the frequency of the main clock source. MAINF carries the new frequency as soon as a low-to-high transition occurs on the MAINFRDY flag.

The measurement is performed on the main frequency (i.e., not limited to RC oscillator only).If the main clock frequency source is the 3 to 20 MHz crystal oscillator, the restart of measurement is not required because of the stability of crystal oscillators.

- **CCSS: Counter Clock Source Selection**

0: The clock of the MAINF counter is the RC oscillator.

1: The clock of the MAINF counter is the crystal oscillator.

Downloaded from Arrow.com.

31.20.10 PMC Clock Generator PLLA Register

Name: CKGR_PLLAR

Address: 0x400E0628

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	ONE	–	–	MULA		

23	22	21	20	19	18	17	16
MULA							

15	14	13	12	11	10	9	8
–	–	PLLACOUNT					

7	6	5	4	3	2	1	0
DIVA							

Possible limitations on PLLA input frequencies and multiplier factors should be checked before using the PMC.

Warning: Bit 29 must always be set to ‘1’ when programming the CKGR_PLLAR register.

This register can only be written if the WPEN bit is cleared in PMC Write Protection Mode Register.

- **DIVA: PLLA Front End Divider**

Value	Name	Description
0	0	Divider output is 0 and PLLA is disabled.
1	BYPASS	Divider is bypassed (divide by 1) and PLLA is enabled.
2–255	–	Divider output is the selected clock divided by DIVA.

- **PLLACOUNT: PLLA Counter**

Specifies the number of slow clock cycles before the LOCKA bit is set in PMC_SR after CKGR_PLLAR is written.

- **MULA: PLLA Multiplier**

0: The PLLA is deactivated (PLLA also disabled if DIVA = 0).

1 up to 62 = The PLLA Clock frequency is the PLLA input frequency multiplied by MULA + 1.

Unlisted values are forbidden.

- **ONE: Must Be Set to 1**

Bit 29 must always be set to ‘1’ when programming the CKGR_PLLAR register.

Downloaded from Arrow.com.

31.20.11 PMC Master Clock Register

Name: PMC_MCKR

Address: 0x400E0630

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	UPLLDIV2	–	–	–	MDIV	
7	6	5	4	3	2	1	0
–	PRES			–	–	CSS	

This register can only be written if the WPEN bit is cleared in PMC Write Protection Mode Register.

- **CSS: Master Clock Source Selection**

Value	Name	Description
0	SLOW_CLK	Slow Clock is selected
1	MAIN_CLK	Main Clock is selected
2	PLLA_CLK	PLLA Clock is selected
3	UPLL_CLK	Divided UPLL Clock is selected

- **PRES: Processor Clock Prescaler**

Value	Name	Description
0	CLK_1	Selected clock
1	CLK_2	Selected clock divided by 2
2	CLK_4	Selected clock divided by 4
3	CLK_8	Selected clock divided by 8
4	CLK_16	Selected clock divided by 16
5	CLK_32	Selected clock divided by 32
6	CLK_64	Selected clock divided by 64
7	CLK_3	Selected clock divided by 3

Downloaded from Arrow.com.

- **MDIV: Master Clock Division**

Value	Name	Description
0	EQ_PCK	Master Clock is Prescaler Output Clock divided by 1.
1	PCK_DIV2	Master Clock is Prescaler Output Clock divided by 2.
2	PCK_DIV4	Master Clock is Prescaler Output Clock divided by 4.
3	PCK_DIV3	Master Clock is Prescaler Output Clock divided by 3.

- **UPLLDIV2: UPLL Divisor by 2**

UPLLDIV2	UPLL Clock Division
0	UPLL clock frequency is divided by 1.
1	UPLL clock frequency is divided by 2.

Downloaded from Arrow.com.

31.20.12 PMC USB Clock Register

Name: PMC_USB

Address: 0x400E0638

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	USBDIV			

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	USBS

This register can only be written if the WPEN bit is cleared in PMC Write Protection Mode Register.

- **USBS: USB Input Clock Selection**

0: USB clock input is PLLA.

1: USB clock input is UPLL.

- **USBDIV: Divider for USB Clock**

USB clock is input clock divided by USBDIV+1.

Downloaded from Arrow.com.

31.20.13 PMC Programmable Clock Register

Name: PMC_PCKx[x=0..6]

Address: 0x400E0640

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	PRES			

7	6	5	4	3	2	1	0
PRES				–	CSS		

This register can only be written if the WPEN bit is cleared in PMC Write Protection Mode Register.

- **CSS: Programmable Clock Source Selection**

Value	Name	Description
0	SLOW_CLK	Slow Clock is selected
1	MAIN_CLK	Main Clock is selected
2	PLLA_CLK	PLLA Clock is selected
3	UPLL_CLK	Divided UPLL Clock is selected
4	MCK	Master Clock is selected

- **PRES: Programmable Clock Prescaler**

0-255: Selected clock is divided by PRES+1.

Downloaded from Arrow.com.

31.20.14 PMC Interrupt Enable Register

Name: PMC_IER

Address: 0x400E0660

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	XT32KERR	–	–	CFDEV	MOSCRCS	MOSCSELS
15	14	13	12	11	10	9	8
–	–	–	–	–	PCKRDY2	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	LOCKU	–	–	MCKRDY	–	LOCKA	MOSCXTS

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

- **MOSCXTS: 3 to 20 MHz Crystal Oscillator Status Interrupt Enable**
- **LOCKA: PLLA Lock Interrupt Enable**
- **MCKRDY: Master Clock Ready Interrupt Enable**
- **LOCKU: UTMI PLL Lock Interrupt Enable**
- **PCKRDYx: Programmable Clock Ready x Interrupt Enable**
- **MOSCSELS: Main Clock Source Oscillator Selection Status Interrupt Enable**
- **MOSCRCS: 4/8/12 MHz RC Oscillator Status Interrupt Enable**
- **CFDEV: Clock Failure Detector Event Interrupt Enable**
- **XT32KERR: 32.768 kHz Crystal Oscillator Error Interrupt Enable**

Downloaded from Arrow.com.

31.20.15 PMC Interrupt Disable Register

Name: PMC_IDR

Address: 0x400E0664

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	XT32KERR	–	–	CFDEV	MOSCRCS	MOSCSELS
15	14	13	12	11	10	9	8
–	–	–	–	–	PCKRDY2	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	LOCKU	–	–	MCKRDY	–	LOCKA	MOSCXTS

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

- **MOSCXTS: 3 to 20 MHz Crystal Oscillator Status Interrupt Disable**
- **LOCKA: PLLA Lock Interrupt Disable**
- **MCKRDY: Master Clock Ready Interrupt Disable**
- **LOCKU: UTMI PLL Lock Interrupt Disable**
- **PCKRDYx: Programmable Clock Ready x Interrupt Disable**
- **MOSCSELS: Main Clock Source Oscillator Selection Status Interrupt Disable**
- **MOSCRCS: 4/8/12 MHz RC Status Interrupt Disable**
- **CFDEV: Clock Failure Detector Event Interrupt Disable**
- **XT32KERR: 32.768 kHz Crystal Oscillator Error Interrupt Disable**

Downloaded from Arrow.com.

31.20.16 PMC Status Register

Name: PMC_SR

Address: 0x400E0668

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	XT32KERR	FOS	CFDS	CFDEV	MOSCRCS	MOSCSELS

15	14	13	12	11	10	9	8
–	–	–	–	–	PCKRDY2	PCKRDY1	PCKRDY0

7	6	5	4	3	2	1	0
OSCSELS	LOCKU	–	–	MCKRDY	–	LOCKA	MOSCXTS

- **MOSCXTS: 3 to 20 MHz Crystal Oscillator Status**

0: 3 to 20 MHz crystal oscillator is not stabilized.

1: 3 to 20 MHz crystal oscillator is stabilized.

- **LOCKA: PLLA Lock Status**

0: PLLA is not locked

1: PLLA is locked.

- **MCKRDY: Master Clock Status**

0: Master Clock is not ready.

1: Master Clock is ready.

- **LOCKU: UTMI PLL Lock Status**

0: UTMI PLL is not locked

1: UTMI PLL is locked.

- **OSCSELS: Slow Clock Source Oscillator Selection**

0: Embedded 32 kHz RC oscillator is selected.

1: 32.768 kHz crystal oscillator is selected.

- **PCKRDYx: Programmable Clock Ready Status**

0: Programmable Clock x is not ready.

1: Programmable Clock x is ready.

- **MOSCSELS: Main Clock Source Oscillator Selection Status**

0: Selection is in progress.

1: Selection is done.

Downloaded from Arrow.com.

- **MOSCRCS: 4/8/12 MHz RC Oscillator Status**

0: 4/8/12 MHz RC oscillator is not stabilized.

1: 4/8/12 MHz RC oscillator is stabilized.

- **CFDEV: Clock Failure Detector Event**

0: No clock failure detection of the 3 to 20 MHz crystal oscillator clock has occurred since the last read of PMC_SR.

1: At least one clock failure detection of the 3 to 20 MHz crystal oscillator clock has occurred since the last read of PMC_SR.

- **CFDS: Clock Failure Detector Status**

0: A clock failure of the 3 to 20 MHz crystal oscillator clock is not detected.

1: A clock failure of the 3 to 20 MHz crystal oscillator clock is detected.

- **FOS: Clock Failure Detector Fault Output Status**

0: The fault output of the clock failure detector is inactive.

1: The fault output of the clock failure detector is active.

- **XT32KERR: Slow Crystal Oscillator Error**

0: The frequency of the 32.768 kHz crystal oscillator is correct (32.768 kHz ±1%) or the monitoring is disabled.

1: The frequency of the 32.768 kHz crystal oscillator is incorrect or has been incorrect for an elapsed period of time since the monitoring has been enabled.

-

Downloaded from Arrow.com.

31.20.17 PMC Interrupt Mask Register

Name: PMC_IMR

Address: 0x400E066C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	XT32KERR	–	–	CFDEV	MOSCRCS	MOSCSELS

15	14	13	12	11	10	9	8
–	–	–	–	–	PCKRDY2	PCKRDY1	PCKRDY0

7	6	5	4	3	2	1	0
–	LOCKU	–	–	MCKRDY	–	LOCKA	MOSCXTS

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

- **MOSCXTS: 3 to 20 MHz Crystal Oscillator Status Interrupt Mask**
- **LOCKA: PLLA Lock Interrupt Mask**
- **MCKRDY: Master Clock Ready Interrupt Mask**
- **LOCKU: UTMI PLL Lock Interrupt Mask**
- **PCKRDYx: Programmable Clock Ready x Interrupt Mask**
- **MOSCSELS: Main Clock Source Oscillator Selection Status Interrupt Mask**
- **MOSCRCS: 4/8/12 MHz RC Status Interrupt Mask**
- **CFDEV: Clock Failure Detector Event Interrupt Mask**
- **XT32KERR: 32.768 kHz Crystal Oscillator Error Interrupt Mask**

Downloaded from Arrow.com.

31.20.18 PMC Fast Startup Mode Register

Name: PMC_FSMR

Address: 0x400E0670

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
FFLPM	FLPM		LPM	–	USBAL	RTCAL	RTTAL

15	14	13	12	11	10	9	8
FSTT15	FSTT14	FSTT13	FSTT12	FSTT11	FSTT10	FSTT9	FSTT8

7	6	5	4	3	2	1	0
FSTT7	FSTT6	FSTT5	FSTT4	FSTT3	FSTT2	FSTT1	FSTT0

This register can only be written if the WPEN bit is cleared in PMC Write Protection Mode Register.

- **FSTT0 - FSTT15: Fast Startup Input Enable 0 to 15**

0: The corresponding wake-up input has no effect on the PMC.

1: The corresponding wake-up input enables a fast restart signal to the PMC.

- **RTTAL: RTT Alarm Enable**

0: The RTT alarm has no effect on the PMC.

1: The RTT alarm enables a fast restart signal to the PMC.

- **RTCAL: RTC Alarm Enable**

0: The RTC alarm has no effect on the PMC.

1: The RTC alarm enables a fast restart signal to the PMC.

- **USBAL: USB Alarm Enable**

0: The USB alarm has no effect on the PMC.

1: The USB alarm enables a fast restart signal to the PMC.

- **LPM: Low-power Mode**

0: The WaitForInterrupt (WFI) or the WaitForEvent (WFE) instruction of the processor makes the processor enter Sleep mode.

1: The WaitForEvent (WFE) instruction of the processor makes the system enter Wait mode.

- **FFLPM: Force Flash Low-power Mode**

0: The Flash Low-power mode, defined in the FLPM field, is automatically applied when in Wait mode and released when going back to Active mode.

1: The Flash Low-power mode is user defined by the FLPM field and immediately applied.

Downloaded from Arrow.com.

- **FLPM: Flash Low-power Mode**

Value	Name	Description
0	FLASH_STANDBY	Flash is in Standby Mode when system enters Wait Mode
1	FLASH_DEEP_POWERDOWN	Flash is in Deep-power-down mode when system enters Wait Mode
2	FLASH_IDLE	Idle mode

Downloaded from Arrow.com.

31.20.19 PMC Fast Startup Polarity Register

Name: PMC_FSPR

Address: 0x400E0674

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
FSTP15	FSTP14	FSTP13	FSTP12	FSTP11	FSTP10	FSTP9	FSTP8
7	6	5	4	3	2	1	0
FSTP7	FSTP6	FSTP5	FSTP4	FSTP3	FSTP2	FSTP1	FSTP0

This register can only be written if the WPEN bit is cleared in PMC Write Protection Mode Register.

- **FSTPx: Fast Startup Input Polarity x**

Defines the active polarity of the corresponding wake-up input. If the corresponding wake-up input is enabled and at the FSTP level, it enables a fast restart signal.

Downloaded from Arrow.com.

31.20.20 PMC Fault Output Clear Register

Name: PMC_FOCR

Address: 0x400E0678

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	FOCLR

- **FOCLR: Fault Output Clear**

Clears the clock failure detector fault output.

Downloaded from Arrow.com.

31.20.21 PMC Write Protection Mode Register

Name: PMC_WPMR

Address: 0x400E06E4

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							

23	22	21	20	19	18	17	16
WPKEY							

15	14	13	12	11	10	9	8
WPKEY							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x504D43 (“PMC” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x504D43 (“PMC” in ASCII).

See Section 31.19 ”Register Write Protection” for the list of registers that can be write-protected.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x504D43	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

Downloaded from Arrow.com.

31.20.22 PMC Write Protection Status Register

Name: PMC_WPSR

Address: 0x400E06E8

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

- **WPVS: Write Protection Violation Status**

0: No write protection violation has occurred since the last read of the PMC_WPSR.

1: A write protection violation has occurred since the last read of the PMC_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protection Violation Source**

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

Downloaded from Arrow.com.

31.20.23 PMC Peripheral Clock Enable Register 1

Name: PMC_PCER1

Address: 0x400E0700

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	PID60	PID59	PID58	PID57	PID56

23	22	21	20	19	18	17	16
–	–	PID53	PID52	PID51	PID50	PID49	PID48

15	14	13	12	11	10	9	8
PID47	PID46	PID45	PID44	PID43	PID42	PID41	PID40

7	6	5	4	3	2	1	0
PID39	–	PID37	–	PID35	PID34	PID33	PID32

This register can only be written if the WPEN bit is cleared in PMC Write Protection Mode Register.

- **PIDx: Peripheral Clock x Enable**

0: No effect.

1: Enables the corresponding peripheral clock.

Notes: 1. PIDx refers to identifiers as defined in the section “Peripheral Identifiers”.

2. Programming the control bits of the Peripheral ID that are not implemented has no effect on the behavior of the PMC.

Downloaded from Arrow.com.

31.20.24 PMC Peripheral Clock Disable Register 1

Name: PMC_PCDR1

Address: 0x400E0704

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	PID60	PID59	PID58	PID57	PID56

23	22	21	20	19	18	17	16
–	–	PID53	PID52	PID51	PID50	PID49	PID48

15	14	13	12	11	10	9	8
PID47	PID46	PID45	PID44	PID43	PID42	PID41	PID40

7	6	5	4	3	2	1	0
PID39	–	PID37	–	PID35	PID34	PID33	PID32

This register can only be written if the WPEN bit is cleared in PMC Write Protection Mode Register.

- **PIDx: Peripheral Clock x Disable**

0: No effect.

1: Disables the corresponding peripheral clock.

Note: PIDx refers to identifiers as defined in the section “Peripheral Identifiers”.

Downloaded from Arrow.com.

31.20.25 PMC Peripheral Clock Status Register 1

Name: PMC_PCSR1

Address: 0x400E0708

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	PID60	PID59	PID58	PID57	PID56

23	22	21	20	19	18	17	16
–	–	PID53	PID52	PID51	PID50	PID49	PID48

15	14	13	12	11	10	9	8
PID47	PID46	PID45	PID44	PID43	PID42	PID41	PID40

7	6	5	4	3	2	1	0
PID39	–	PID37	–	PID35	PID34	PID33	PID32

- **PIDx: Peripheral Clock x Status**

0: The corresponding peripheral clock is disabled.

1: The corresponding peripheral clock is enabled.

Note: PIDx refers to identifiers as defined in the section “Peripheral Identifiers”.

Downloaded from Arrow.com.

31.20.26 PMC Peripheral Control Register

Name: PMC_PCR

Address: 0x400E070C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	GCLKEN	EN	GCLKDIV			

23	22	21	20	19	18	17	16
GCLKDIV				–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	CMD	–	GCLKCSS		

7	6	5	4	3	2	1	0
–	PID						

- **PID: Peripheral ID**

Peripheral ID selection from PID2 to PID127.

PID2 to PID127 refer to identifiers as defined in Section 13.1 ”Peripheral Identifiers”.

- **GCLKCSS: Generic Clock Source Selection**

Value	Name	Description
0	SLOW_CLK	Slow clock is selected
1	MAIN_CLK	Main clock is selected
2	PLLA_CLK	PLLACK is selected
3	UPLL_CLK	UPLL Clock is selected
4	MCK_CLK	Master Clock is selected

- **CMD: Command**

0: Read mode.

1: Write mode.

- **GCLKDIV: Generic Clock Division Ratio**

Generic clock is the selected clock period divided by GCLKDIV + 1.

GCLKDIV must not be changed while the peripheral selects GCLKx (e.g., bit rate, etc.).

- **EN: Enable**

0: Selected Peripheral clock is disabled.

1: Selected Peripheral clock is enabled.

- **GCLKEN: Generic Clock Enable**

0: The selected generic clock is disabled.

1: The selected generic clock is enabled.

Downloaded from Arrow.com.

31.20.27 PMC Oscillator Calibration Register

Name: PMC_OCR

Address: 0x400E0710

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
SEL12	CAL12						
15	14	13	12	11	10	9	8
SEL8	CAL8						
7	6	5	4	3	2	1	0
SEL4	CAL4						

This register can only be written if the WPEN bit is cleared in PMC Write Protection Mode Register.

- **CAL4: RC Oscillator Calibration bits for 4 MHz**

Calibration bits applied to the RC Oscillator when SEL4 is set.

- **SEL4: Selection of RC Oscillator Calibration bits for 4 MHz**

0: Default value stored in Flash memory.

1: Value written by user in CAL4 field of this register.

- **CAL8: RC Oscillator Calibration bits for 8 MHz**

Calibration bits applied to the RC Oscillator when SEL8 is set.

- **SEL8: Selection of RC Oscillator Calibration bits for 8 MHz**

0: Factory-determined value stored in Flash memory.

1: Value written by user in CAL8 field of this register.

- **CAL12: RC Oscillator Calibration bits for 12 MHz**

Calibration bits applied to the RC Oscillator when SEL12 is set.

- **SEL12: Selection of RC Oscillator Calibration bits for 12 MHz**

0: Factory-determined value stored in Flash memory.

1: Value written by user in CAL12 field of this register.

Downloaded from Arrow.com.

31.20.28 PMC SleepWalking Enable Register 0

Name: PMC_SLPWK_ER0

Address: 0x400E0714

Access: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	–	–	–	–	–	–	–

This register can only be written if the WPEN bit is cleared in PMC Write Protection Mode Register.

- **PIDx: Peripheral x SleepWalking Enable**

0: No effect.

1: The asynchronous partial wake-up (SleepWalking) function of the corresponding peripheral is enabled.

Not all PIDs can be configured with asynchronous partial wake-up.

Only the following PID can be configured with asynchronous partial wake-up: UARTx and TWIHSx.

The clock of the peripheral must be enabled before using its asynchronous partial wake-up (SleepWalking) function (its associated PIDx field in PMC Peripheral Clock Status Register 0 or PMC Peripheral Clock Status Register 1 is set to ‘1’).

Note: PIDx refers to identifiers as defined in the section “Peripheral Identifiers”.

Downloaded from Arrow.com.

31.20.29 PMC SleepWalking Enable Register 1

Name: PMC_SLPWK_ER1

Address: 0x400E0734

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	PID60	PID59	PID58	PID57	PID56
23	22	21	20	19	18	17	16
–	–	PID53	PID52	PID51	PID50	PID49	PID48
15	14	13	12	11	10	9	8
PID47	PID46	PID45	PID44	PID43	PID42	PID41	PID40
7	6	5	4	3	2	1	0
PID39	–	PID37	–	PID35	PID34	PID33	PID32

This register can only be written if the WPEN bit is cleared in PMC Write Protection Mode Register.

- **PIDx: Peripheral x SleepWalking Enable**

0: No effect.

1: The asynchronous partial wake-up (SleepWalking) function of the corresponding peripheral is enabled.

Not all PIDs can be configured with asynchronous partial wake-up.

Only the following PID can be configured with asynchronous partial wake-up: UARTx and TWIHSx.

The clock of the peripheral must be enabled before using its asynchronous partial wake-up (SleepWalking) function (the associated PIDx field in PMC Peripheral Clock Status Register 1 or PMC Peripheral Clock Status Register 0 is set to ‘1’).

Note: PIDx refers to identifiers as defined in the section “Peripheral Identifiers”.

Downloaded from Arrow.com.

31.20.30 PMC SleepWalking Disable Register 0

Name: PMC_SLPWK_DR0

Address: 0x400E0718

Access: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	–	–	–	–	–	–	–

This register can only be written if the WPEN bit is cleared in PMC Write Protection Mode Register.

- **PIDx: Peripheral x SleepWalking Disable**

0: No effect.

1: The asynchronous partial wake-up (SleepWalking) function of the corresponding peripheral is disabled.

Not all PIDs can be configured with asynchronous partial wake-up.

Only the following PIDs can be configured with asynchronous partial wake-up: UARTx and TWIHSx.

Note: PIDx refers to identifiers as defined in the section “Peripheral Identifiers”.

Downloaded from Arrow.com.

31.20.31 PMC SleepWalking Disable Register 1

Name: PMC_SLPWK_DR1

Address: 0x400E0738

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	PID60	PID59	PID58	PID57	PID56
23	22	21	20	19	18	17	16
–	–	PID53	PID52	PID51	PID50	PID49	PID48
15	14	13	12	11	10	9	8
PID47	PID46	PID45	PID44	PID43	PID42	PID41	PID40
7	6	5	4	3	2	1	0
PID39	–	PID37	–	PID35	PID34	PID33	PID32

This register can only be written if the WPEN bit is cleared in PMC Write Protection Mode Register.

- **PIDx: Peripheral x SleepWalking Disable**

0: No effect.

1: The asynchronous partial wake-up (SleepWalking) function of the corresponding peripheral is disabled.

Not all PIDs can be configured with asynchronous partial wake-up.

Only the following PIDs can be configured with asynchronous partial wake-up: UARTx and TWIHSx.

Note: PIDx refers to identifiers as defined in the section “Peripheral Identifiers”.

Downloaded from Arrow.com.

31.20.32 PMC SleepWalking Status Register 0

Name: PMC_SLPWK_SR0

Address: 0x400E071C

Access: Read-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	–	–	–	–	–	–	–

- **PIDx: Peripheral x SleepWalking Status**

0: The asynchronous partial wake-up (SleepWalking) function of the peripheral is currently disabled or the peripheral enabled for asynchronous partial wake-up (SleepWalking) cleared the PIDx bit upon detection of a wake-up condition.

1: The asynchronous partial wake-up (SleepWalking) function of the peripheral is currently enabled.

Not all PIDs can be configured with asynchronous partial wake-up.

Only the following PIDs can be configured with asynchronous partial wake-up: UARTx and TWIHSx.

Note: PIDx refers to identifiers as defined in the section “Peripheral Identifiers”.

Downloaded from Arrow.com.

31.20.33 PMC SleepWalking Status Register 1

Name: PMC_SLPWK_SR1

Address: 0x400E073C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	PID60	PID59	PID58	PID57	PID56
23	22	21	20	19	18	17	16
–	–	PID53	PID52	PID51	PID50	PID49	PID48
15	14	13	12	11	10	9	8
PID47	PID46	PID45	PID44	PID43	PID42	PID41	PID40
7	6	5	4	3	2	1	0
PID39	–	PID37	–	PID35	PID34	PID33	PID32

- **PIDx: Peripheral x SleepWalking Status**

0: The asynchronous partial wake-up (SleepWalking) function of the peripheral is currently disabled or the peripheral enabled for asynchronous partial wake-up (SleepWalking) cleared the PIDx bit upon detection of a wake-up condition.

1: The asynchronous partial wake-up (SleepWalking) function of the peripheral is currently enabled.

Not all PIDs can be configured with asynchronous partial wake-up.

Only the following PIDs can be configured with asynchronous partial wake-up: UARTx and TWIHSx.

Note: PIDx refers to identifiers as defined in the section “Peripheral Identifiers”.

Downloaded from Arrow.com.

31.20.34 PMC SleepWalking Activity Status Register 0

Name: PMC_SLPWK_ASR0

Address: 0x400E0720

Access: Read-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	–	–	–	–	–	–	–

- **PIDx: Peripheral x Activity Status**

0: The peripheral x is not currently active. The asynchronous partial wake-up (SleepWalking) function can be activated.

1: The peripheral x is currently active. The asynchronous partial wake-up (SleepWalking) function must not be activated.

Only the following PIDs can be configured with asynchronous partial wake-up: UARTx and TWIHSx.
All other PIDs are always read at 0.

Note: PIDx refers to identifiers as defined in the section “Peripheral Identifiers”.

Downloaded from Arrow.com.

31.20.35 PLL Maximum Multiplier Value Register

Name: PMC_PMMR

Address: 0x400E0730

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	PLLA_MMAX		
7	6	5	4	3	2	1	0
PLLA_MMAX							

This register can only be written if the WPEN bit is cleared in the PMC Write Protection Mode Register.

- **PLLA_MMAX: PLLA Maximum Allowed Multiplier Value**

Defines the maximum value of multiplication factor that can be sent to PLLA. Any value of the MULA field (see PMC Clock Generator PLLA Register) above PLLA_MMAX is saturated to PLLA_MMAX. PLLA_MMAX write operation is cancelled in the following cases:

- The value of MULA is currently saturated by PLLA_MMAX
- The user is trying to write a value of PLLA_MMAX that is smaller than the current value of MULA

Downloaded from Arrow.com.

31.20.36 PMC SleepWalking Activity Status Register 1

Name: PMC_SLPWK_ASR1

Address: 0x400E0740

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	PID60	PID59	PID58	PID57	PID56
23	22	21	20	19	18	17	16
–	–	PID53	PID52	PID51	PID50	PID49	PID48
15	14	13	12	11	10	9	8
PID47	PID46	PID45	PID44	PID43	PID42	PID41	PID40
7	6	5	4	3	2	1	0
PID39	–	PID37	–	PID35	PID34	PID33	PID32

- **PIDx: Peripheral x Activity Status**

0: The peripheral x is not currently active. The asynchronous partial wake-up (SleepWalking) function can be activated.

1: The peripheral x is currently active. The asynchronous partial wake-up (SleepWalking) function must not be activated.

Only the following PIDs can be configured with asynchronous partial wake-up: UARTx and TWIHSx.
All other PIDs are always read at 0.

Note: PIDx refers to identifiers as defined in the section “Peripheral Identifiers”.

Downloaded from Arrow.com.

31.20.37 PMC SleepWalking Activity In Progress Register

Name: PMC_SLPWK_AIPR

Address: 0x400E0744

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	AIP

- **AIP: Activity In Progress**

0: There is no activity on peripherals. The asynchronous partial wake-up (SleepWalking) function can be activated on one or more peripherals. The device can enter Wait mode.

Only the following PIDs can be configured with asynchronous partial wake-up: UARTx and TWIHSx.

1: One or more peripherals are currently active. The device must not enter Wait mode if the asynchronous partial wake-up is enabled for one of the following PIDs: UARTx and TWIHSx.

Downloaded from Arrow.com.

32. Parallel Input/Output Controller (PIO)

32.1 Description

The Parallel Input/Output Controller (PIO) manages up to 32 fully programmable input/output lines. Each I/O line may be dedicated as a general-purpose I/O or be assigned to a function of an embedded peripheral. This ensures effective optimization of the pins of the product.

Each I/O line is associated with a bit number in all of the 32-bit registers of the 32-bit wide user interface.

Each I/O line of the PIO Controller features:

- An input change interrupt enabling level change detection on any I/O line.
- Additional Interrupt modes enabling rising edge, falling edge, low-level or high-level detection on any I/O line.
- A glitch filter providing rejection of glitches lower than one-half of peripheral clock cycle.
- A debouncing filter providing rejection of unwanted pulses from key or push button operations.
- Multi-drive capability similar to an open drain I/O line.
- Control of the pull-up and pull-down of the I/O line.
- Input visibility and output control.

The PIO Controller also features a synchronous output providing up to 32 bits of data output in a single write operation.

An 8-bit parallel capture mode is also available which can be used to interface a CMOS digital image sensor, an ADC, a DSP synchronous port in synchronous mode, etc.

Downloaded from Arrow.com.

32.2 Embedded Characteristics

- Up to 32 Programmable I/O Lines
- Fully Programmable through Set/Clear Registers
- Multiplexing of Four Peripheral Functions per I/O Line
- For each I/O Line (Whether Assigned to a Peripheral or Used as General Purpose I/O)
  - Input Change Interrupt
  - Programmable Glitch Filter
  - Programmable Debouncing Filter
  - Multi-drive Option Enables Driving in Open Drain
  - Programmable Pull-Up on Each I/O Line
  - Pin Data Status Register, Supplies Visibility of the Level on the Pin at Any Time
  - Additional Interrupt Modes on a Programmable Event: Rising Edge, Falling Edge, Low-Level or High-Level
  - Lock of the Configuration by the Connected Peripheral
- Synchronous Output, Provides Set and Clear of Several I/O Lines in a Single Write
- Register Write Protection
- Programmable Schmitt Trigger Inputs
- Programmable I/O Drive
- Parallel Capture Mode
  - Can Be Used to Interface a CMOS Digital Image Sensor, an ADC, etc.
  - One Clock, 8-bit Parallel Data and Two Data Enable on I/O Lines
  - Data Can be Sampled Every Other Time (For Chrominance Sampling Only)
  - Supports Connection of One DMA Controller Channel Which Offers Buffer Reception Without Processor Intervention

Downloaded from Arrow.com.

32.3 Block Diagram

Figure 32-1. Block Diagram

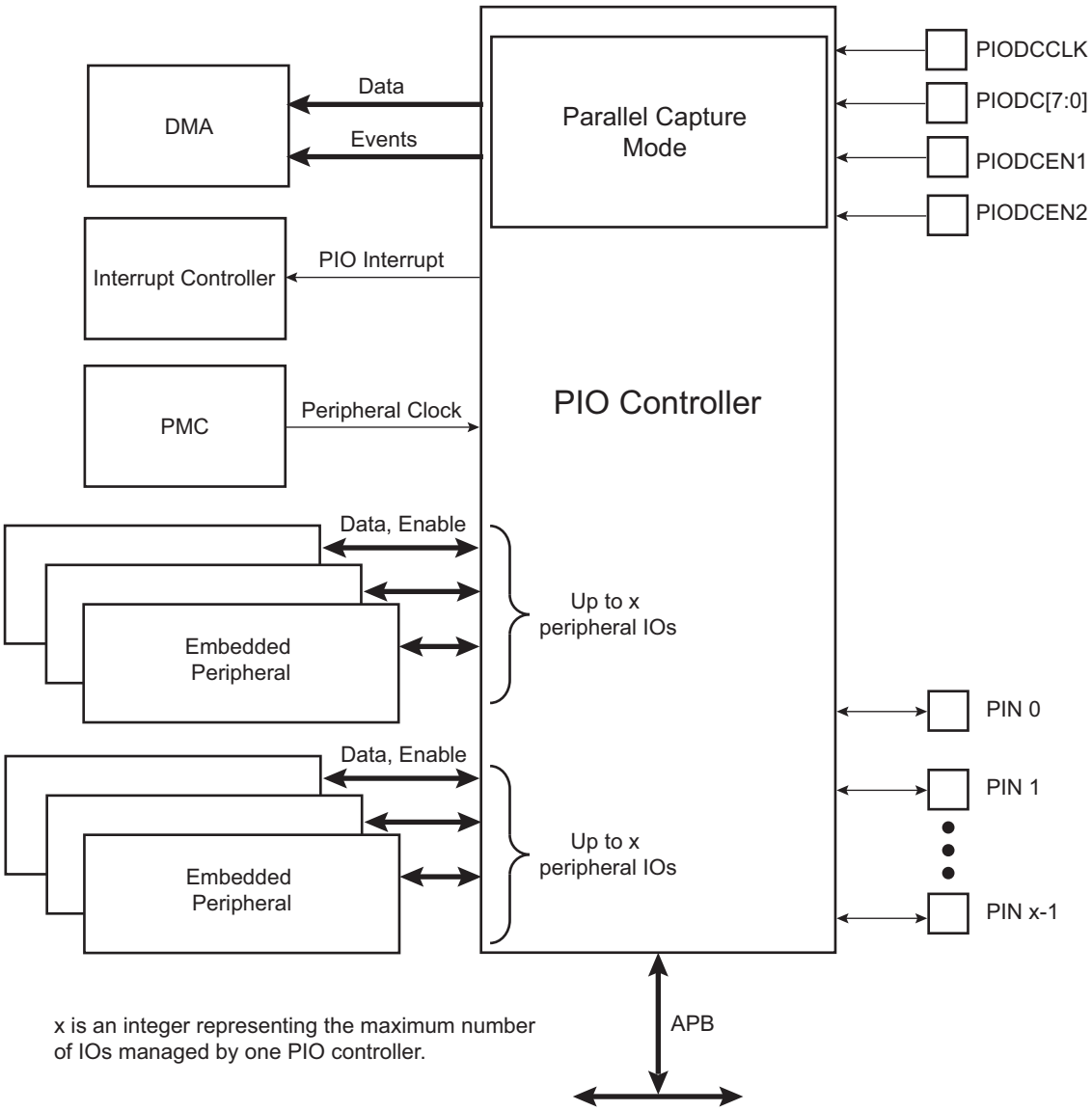


Table 32-1. Signal Description

Signal Name	Signal Description	Signal Type
PIODCCLK	Parallel Capture Mode Clock	Input
PIODC[7:0]	Parallel Capture Mode Data	Input
PIODCEN1	Parallel Capture Mode Data Enable 1	Input
PIODCEN2	Parallel Capture Mode Data Enable 2	Input

Downloaded from Arrow.com.

32.4 Product Dependencies

32.4.1 Pin Multiplexing

Each pin is configurable, depending on the product, as either a general-purpose I/O line only, or as an I/O line multiplexed with one or two peripheral I/Os. As the multiplexing is hardware defined and thus product-dependent, the hardware designer and programmer must carefully determine the configuration of the PIO Controllers required by their application. When an I/O line is general-purpose only, i.e., not multiplexed with any peripheral I/O, programming of the PIO Controller regarding the assignment to a peripheral has no effect and only the PIO Controller can control how the pin is driven by the product.

32.4.2 External Interrupt Lines

When the WKUPx input pins must be used as external interrupt lines, the PIO Controller must be configured to disable the peripheral control on these IOs, and the corresponding IO lines must be set to Input mode.

32.4.3 Power Management

The Power Management Controller controls the peripheral clock in order to save power. Writing any of the registers of the user interface does not require the peripheral clock to be enabled. This means that the configuration of the I/O lines does not require the peripheral clock to be enabled.

However, when the clock is disabled, not all of the features of the PIO Controller are available, including glitch filtering. Note that the input change interrupt, the interrupt modes on a programmable event and the read of the pin level require the clock to be validated.

After a hardware reset, the peripheral clock is disabled by default.

The user must configure the Power Management Controller before any access to the input line information.

32.4.4 Interrupt Sources

For interrupt handling, the PIO Controllers are considered as user peripherals. This means that the PIO Controller interrupt lines are connected among the interrupt sources. Refer to the PIO Controller peripheral identifier in Section 13.1 ”Peripheral Identifiers” to identify the interrupt sources dedicated to the PIO Controllers. Using the PIO Controller requires the Interrupt Controller to be programmed first.

The PIO Controller interrupt can be generated only if the peripheral clock is enabled.

Table 32-2. Peripheral IDs

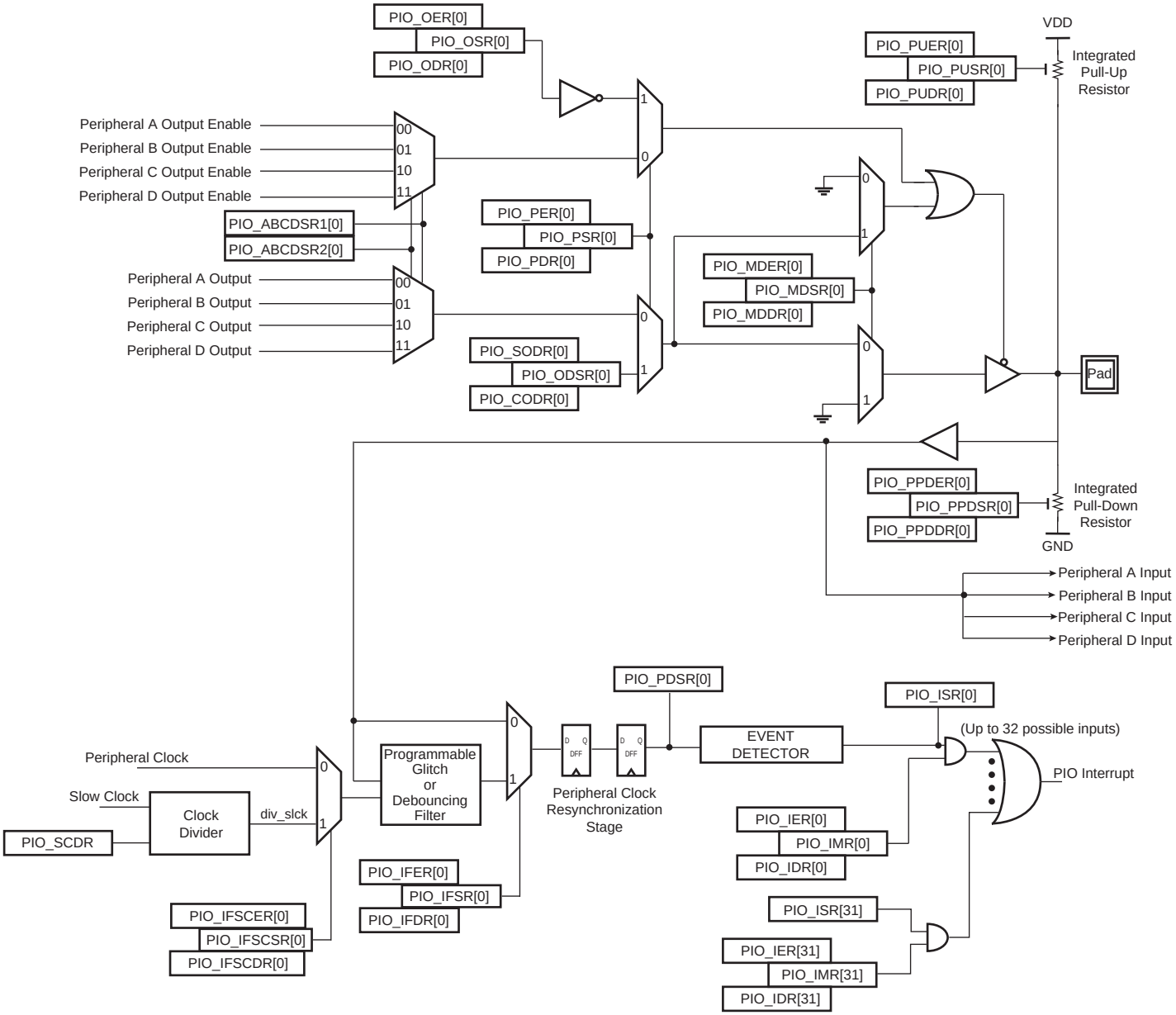
Instance	ID
PIOA	10
PIOB	11
PIOC	12
PIOD	16
PIOE	17

Downloaded from Arrow.com.

32.5 Functional Description

The PIO Controller features up to 32 fully-programmable I/O lines. Most of the control logic associated to each I/O is represented in Figure 32-2. In this description each signal shown represents one of up to 32 possible indexes.

Figure 32-2. I/O Line Control Logic



32.5.1 Pull-up and Pull-down Resistor Control

Each I/O line is designed with an embedded pull-up resistor and an embedded pull-down resistor. The pull-up resistor can be enabled or disabled by writing to the Pull-up Enable Register (PIO_PUER) or Pull-up Disable Register (PIO_PUDR), respectively. Writing to these registers results in setting or clearing the corresponding bit in the Pull-up Status Register (PIO_PUSR). Reading a one in PIO_PUSR means the pull-up is disabled and reading a zero means the pull-up is enabled. The pull-down resistor can be enabled or disabled by writing the Pull-down Enable Register (PIO_PPDER) or the Pull-down Disable Register (PIO_PPDDR), respectively. Writing in these

Downloaded from Arrow.com.

registers results in setting or clearing the corresponding bit in the Pull-down Status Register (PIO_PPDSR). Reading a one in PIO_PPDSR means the pull-up is disabled and reading a zero means the pull-down is enabled.

Enabling the pull-down resistor while the pull-up resistor is still enabled is not possible. In this case, the write of PIO_PPDER for the relevant I/O line is discarded. Likewise, enabling the pull-up resistor while the pull-down resistor is still enabled is not possible. In this case, the write of PIO_PUER for the relevant I/O line is discarded.

Control of the pull-up resistor is possible regardless of the configuration of the I/O line.

After reset, depending on the I/O, pull-up or pull-down can be set.

32.5.2 I/O Line or Peripheral Function Selection

When a pin is multiplexed with one or two peripheral functions, the selection is controlled with the Enable Register (PIO_PER) and the Disable Register (PIO_PDR). The Status Register (PIO_PSR) is the result of the set and clear registers and indicates whether the pin is controlled by the corresponding peripheral or by the PIO Controller. A value of zero indicates that the pin is controlled by the corresponding on-chip peripheral selected in the ABCD Select registers (PIO_ABCDSR1 and PIO_ABCDSR2). A value of one indicates the pin is controlled by the PIO Controller.

If a pin is used as a general-purpose I/O line (not multiplexed with an on-chip peripheral), PIO_PER and PIO_PDR have no effect and PIO_PSR returns a one for the corresponding bit.

After reset, the I/O lines are controlled by the PIO Controller, i.e., PIO_PSR resets at one. However, in some events, it is important that PIO lines are controlled by the peripheral (as in the case of memory chip select lines that must be driven inactive after reset, or for address lines that must be driven low for booting out of an external memory). Thus, the reset value of PIO_PSR is defined at the product level and depends on the multiplexing of the device.

32.5.3 Peripheral A or B or C or D Selection

The PIO Controller provides multiplexing of up to four peripheral functions on a single pin. The selection is performed by writing PIO_ABCDSR1 and PIO_ABCDSR2.

For each pin:

- The corresponding bit at level zero in PIO_ABCDSR1 and the corresponding bit at level zero in PIO_ABCDSR2 means peripheral A is selected.
- The corresponding bit at level one in PIO_ABCDSR1 and the corresponding bit at level zero in PIO_ABCDSR2 means peripheral B is selected.
- The corresponding bit at level zero in PIO_ABCDSR1 and the corresponding bit at level one in PIO_ABCDSR2 means peripheral C is selected.
- The corresponding bit at level one in PIO_ABCDSR1 and the corresponding bit at level one in PIO_ABCDSR2 means peripheral D is selected.

Note that multiplexing of peripheral lines A, B, C and D only affects the output line. The peripheral input lines are always connected to the pin input (see Figure 32-2).

Writing in PIO_ABCDSR1 and PIO_ABCDSR2 manages the multiplexing regardless of the configuration of the pin. However, assignment of a pin to a peripheral function requires a write in PIO_ABCDSR1 and PIO_ABCDSR2 in addition to a write in PIO_PDR.

After reset, PIO_ABCDSR1 and PIO_ABCDSR2 are zero, thus indicating that all the PIO lines are configured on peripheral A. However, peripheral A generally does not drive the pin as the PIO Controller resets in I/O line mode.

If the software selects a peripheral A, B, C or D which does not exist for a pin, no alternate functions are enabled for this pin and the selection is taken into account. The PIO Controller does not carry out checks to prevent selection of a peripheral which does not exist.

Downloaded from Arrow.com.

32.5.4 Output Control

When the I/O line is assigned to a peripheral function, i.e., the corresponding bit in PIO_PSR is at zero, the drive of the I/O line is controlled by the peripheral. Peripheral A or B or C or D depending on the value in PIO_ABCDSR1 and PIO_ABCDSR2 determines whether the pin is driven or not.

When the I/O line is controlled by the PIO Controller, the pin can be configured to be driven. This is done by writing the Output Enable Register (PIO_OER) and Output Disable Register (PIO_ODR). The results of these write operations are detected in the Output Status Register (PIO_OSR). When a bit in this register is at zero, the corresponding I/O line is used as an input only. When the bit is at one, the corresponding I/O line is driven by the PIO Controller.

The level driven on an I/O line can be determined by writing in the Set Output Data Register (PIO_SODR) and the Clear Output Data Register (PIO_CODR). These write operations, respectively, set and clear the Output Data Status Register (PIO_ODSR), which represents the data driven on the I/O lines. Writing in PIO_OER and PIO_ODR manages PIO_OSR whether the pin is configured to be controlled by the PIO Controller or assigned to a peripheral function. This enables configuration of the I/O line prior to setting it to be managed by the PIO Controller.

Similarly, writing in PIO_SODR and PIO_CODR affects PIO_ODSR. This is important as it defines the first level driven on the I/O line.

32.5.5 Synchronous Data Output

Clearing one or more PIO line(s) and setting another one or more PIO line(s) synchronously cannot be done by using PIO_SODR and PIO_CODR. It requires two successive write operations into two different registers. To overcome this, the PIO Controller offers a direct control of PIO outputs by single write access to PIO_ODSR. Only bits unmasked by the Output Write Status Register (PIO_OWSR) are written. The mask bits in PIO_OWSR are set by writing to the Output Write Enable Register (PIO_OWER) and cleared by writing to the Output Write Disable Register (PIO_OWDR).

After reset, the synchronous data output is disabled on all the I/O lines as PIO_OWSR resets at 0x0.

32.5.6 Multi-Drive Control (Open Drain)

Each I/O can be independently programmed in open drain by using the multi-drive feature. This feature permits several drivers to be connected on the I/O line which is driven low only by each device. An external pull-up resistor (or enabling of the internal one) is generally required to guarantee a high level on the line.

The multi-drive feature is controlled by the Multi-driver Enable Register (PIO_MDER) and the Multi-driver Disable Register (PIO_MDDR). The multi-drive can be selected whether the I/O line is controlled by the PIO Controller or assigned to a peripheral function. The Multi-driver Status Register (PIO_MDSR) indicates the pins that are configured to support external drivers.

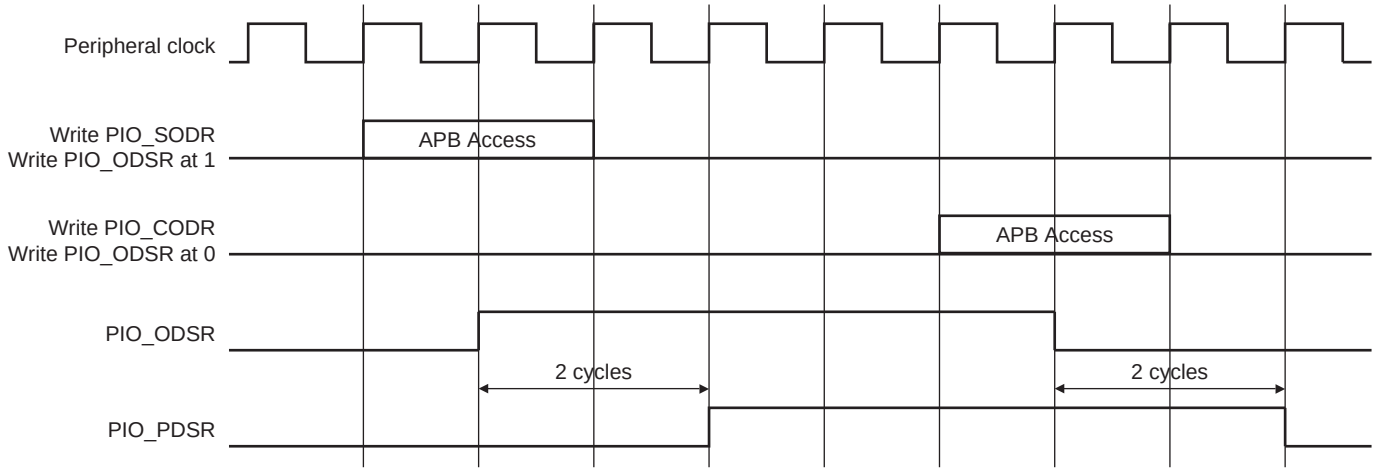
After reset, the multi-drive feature is disabled on all pins, i.e., PIO_MDSR resets at value 0x0.

32.5.7 Output Line Timings

Figure 32-3 shows how the outputs are driven either by writing PIO_SODR or PIO_CODR, or by directly writing PIO_ODSR. This last case is valid only if the corresponding bit in PIO_OWSR is set. Figure 32-3 also shows when the feedback in the Pin Data Status Register (PIO_PDSR) is available.

Downloaded from Arrow.com.

Figure 32-3. Output Line Timings



32.5.8 Inputs

The level on each I/O line can be read through PIO_PDSR. This register indicates the level of the I/O lines regardless of their configuration, whether uniquely as an input, or driven by the PIO Controller, or driven by a peripheral.

Reading the I/O line levels requires the clock of the PIO Controller to be enabled, otherwise PIO_PDSR reads the levels present on the I/O line at the time the clock was disabled.

32.5.9 Input Glitch and Debouncing Filters

Optional input glitch and debouncing filters are independently programmable on each I/O line.

The glitch filter can filter a glitch with a duration of less than 1/2 peripheral clock and the debouncing filter can filter a pulse of less than 1/2 period of a programmable divided slow clock.

The selection between glitch filtering or debounce filtering is done by writing in the PIO Input Filter Slow Clock Disable Register (PIO_IFSCDR) and the PIO Input Filter Slow Clock Enable Register (PIO_IFSCER). Writing PIO_IFSCDR and PIO_IFSCER, respectively, sets and clears bits in the Input Filter Slow Clock Status Register (PIO_IFSCSR).

The current selection status can be checked by reading the PIO_IFSCSR.

- If PIO_IFSCSR[i] = 0: The glitch filter can filter a glitch with a duration of less than 1/2 master clock period.
- If PIO_IFSCSR[i] = 1: The debouncing filter can filter a pulse with a duration of less than 1/2 programmable divided slow clock period.

For the debouncing filter, the period of the divided slow clock is defined by writing in the DIV field of the Slow Clock Divider Debouncing Register (PIO_SCDR):

$t_{div_slck} = ((DIV + 1) \times 2) \times t_{slck}$

When the glitch or debouncing filter is enabled, a glitch or pulse with a duration of less than 1/2 selected clock cycle (selected clock represents peripheral clock or divided slow clock depending on PIO_IFSCDR and PIO_IFSCER programming) is automatically rejected, while a pulse with a duration of one selected clock (peripheral clock or divided slow clock) cycle or more is accepted. For pulse durations between 1/2 selected clock cycle and one selected clock cycle, the pulse may or may not be taken into account, depending on the precise timing of its occurrence. Thus for a pulse to be visible, it must exceed one selected clock cycle, whereas for a glitch to be reliably filtered out, its duration must not exceed 1/2 selected clock cycle.

The filters also introduce some latencies, illustrated in Figure 32-4 and Figure 32-5.

Downloaded from Arrow.com.

The glitch filters are controlled by the Input Filter Enable Register (PIO_IFER), the Input Filter Disable Register (PIO_IFDR) and the Input Filter Status Register (PIO_IFSR). Writing PIO_IFER and PIO_IFDR respectively sets and clears bits in PIO_IFSR. This last register enables the glitch filter on the I/O lines.

When the glitch and/or debouncing filter is enabled, it does not modify the behavior of the inputs on the peripherals. It acts only on the value read in PIO_PDSR and on the input change interrupt detection. The glitch and debouncing filters require that the peripheral clock is enabled.

Figure 32-4. Input Glitch Filter Timing

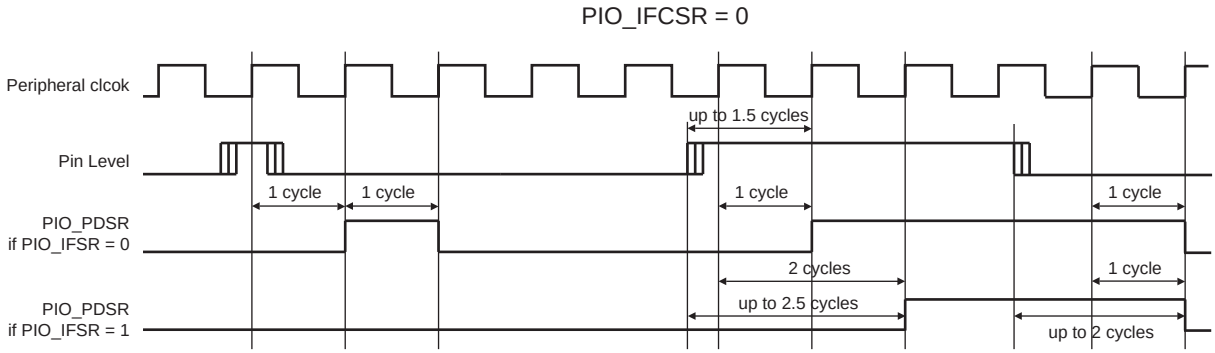
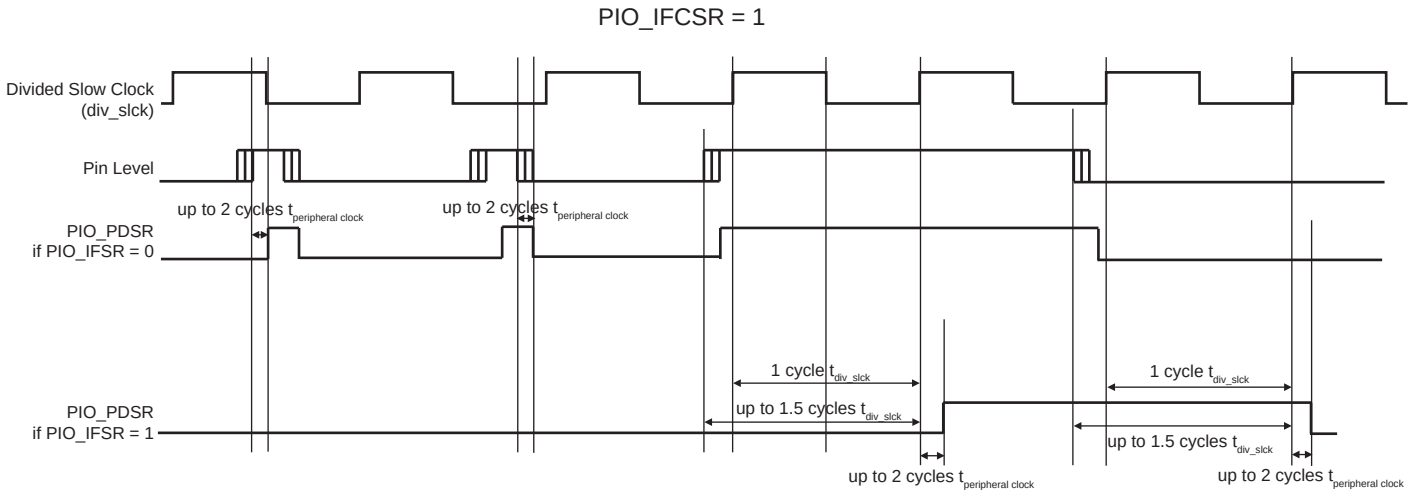


Figure 32-5. Input Debouncing Filter Timing



32.5.10 Input Edge/Level Interrupt

The PIO Controller can be programmed to generate an interrupt when it detects an edge or a level on an I/O line. The Input Edge/Level interrupt is controlled by writing the Interrupt Enable Register (PIO_IER) and the Interrupt Disable Register (PIO_IDR), which enable and disable the input change interrupt respectively by setting and clearing the corresponding bit in the Interrupt Mask Register (PIO_IMR). As input change detection is possible only by comparing two successive samplings of the input of the I/O line, the peripheral clock must be enabled. The Input Change interrupt is available regardless of the configuration of the I/O line, i.e., configured as an input only, controlled by the PIO Controller or assigned to a peripheral function.

By default, the interrupt can be generated at any time an edge is detected on the input.

Some additional interrupt modes can be enabled/disabled by writing in the Additional Interrupt Modes Enable Register (PIO_AIMER) and Additional Interrupt Modes Disable Register (PIO_AIMDR). The current state of this selection can be read through the Additional Interrupt Modes Mask Register (PIO_AIMMR).

These additional modes are:

Downloaded from Arrow.com.

- Rising edge detection
- Falling edge detection
- Low-level detection
- High-level detection

In order to select an additional interrupt mode:

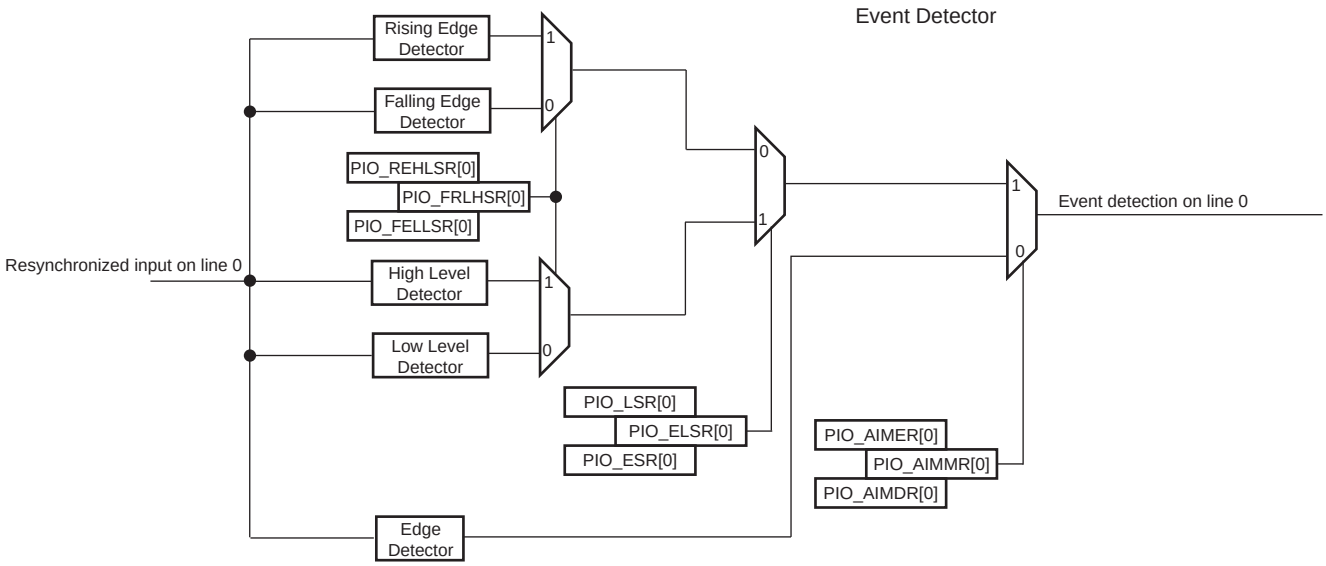
- The type of event detection (edge or level) must be selected by writing in the Edge Select Register (PIO_ESR) and Level Select Register (PIO_LSR) which select, respectively, the edge and level detection. The current status of this selection is accessible through the Edge/Level Status Register (PIO_ELSR).
- The polarity of the event detection (rising/falling edge or high/low-level) must be selected by writing in the Falling Edge/Low-Level Select Register (PIO_FELLSR) and Rising Edge/High-Level Select Register (PIO_REHLSR) which allow to select falling or rising edge (if edge is selected in PIO_ELSR) edge or high- or low-level detection (if level is selected in PIO_ELSR). The current status of this selection is accessible through the Fall/Rise - Low/High Status Register (PIO_FRLHSR).

When an input edge or level is detected on an I/O line, the corresponding bit in the Interrupt Status Register (PIO_ISR) is set. If the corresponding bit in PIO_IMR is set, the PIO Controller interrupt line is asserted.The interrupt signals of the 32 channels are ORed-wired together to generate a single interrupt signal to the interrupt controller.

When the software reads PIO_ISR, all the interrupts are automatically cleared. This signifies that all the interrupts that are pending when PIO_ISR is read must be handled. When an Interrupt is enabled on a “level”, the interrupt is generated as long as the interrupt source is not cleared, even if some read accesses in PIO_ISR are performed.

Downloaded from Arrow.com.

Figure 32-6. Event Detector on Input Lines (Figure Represents Line 0)



Example of interrupt generation on following lines:

- Rising edge on PIO line 0
- Falling edge on PIO line 1
- Rising edge on PIO line 2
- Low-level on PIO line 3
- High-level on PIO line 4
- High-level on PIO line 5
- Falling edge on PIO line 6
- Rising edge on PIO line 7
- Any edge on the other lines

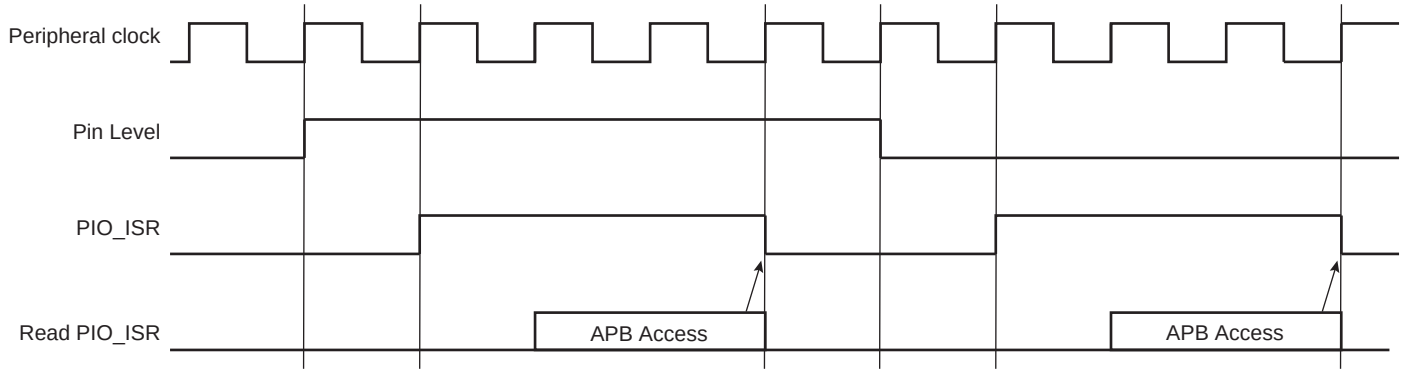
Table 32-3 provides the required configuration for this example.

Table 32-3. Configuration for Example Interrupt Generation

Configuration	Description
Interrupt Mode	All the interrupt sources are enabled by writing 32'hFFFF_FFFF in PIO_IER. Then the additional interrupt mode is enabled for lines 0 to 7 by writing 32'h0000_00FF in PIO_AIMER.
Edge or Level Detection	Lines 3, 4 and 5 are configured in level detection by writing 32'h0000_0038 in PIO_LSR. The other lines are configured in edge detection by default, if they have not been previously configured. Otherwise, lines 0, 1, 2, 6 and 7 must be configured in edge detection by writing 32'h0000_00C7 in PIO_ESR.
Falling/Rising Edge or Low/High-Level Detection	Lines 0, 2, 4, 5 and 7 are configured in rising edge or high-level detection by writing 32'h0000_00B5 in PIO_REHLSR. The other lines are configured in falling edge or low-level detection by default if they have not been previously configured. Otherwise, lines 1, 3 and 6 must be configured in falling edge/low-level detection by writing 32'h0000_004A in PIO_FELLSR.

Downloaded from Arrow.com.

Figure 32-7. Input Change Interrupt Timings When No Additional Interrupt Modes



32.5.11 I/O Lines Lock

When an I/O line is controlled by a peripheral (particularly the Pulse Width Modulation Controller PWM), it can become locked by the action of this peripheral via an input of the PIO Controller. When an I/O line is locked, the write of the corresponding bit in PIO_PER, PIO_PDR, PIO_MDER, PIO_MDDR, PIO_PUDR, PIO_PUER, PIO_ABCDSR1 and PIO_ABCDSR2 is discarded in order to lock its configuration. The user can know at anytime which I/O line is locked by reading the PIO Lock Status Register (PIO_LOCKSR). Once an I/O line is locked, the only way to unlock it is to apply a hardware reset to the PIO Controller.

32.5.12 Programmable I/O Drive

It is possible to configure the I/O drive for pads PA0-PA31, PB0-PB13, PC0-PC31, PD0-PD31 and PE0-PE5.. Refer to Section 56. “Electrical Characteristics”.

32.5.13 Programmable Schmitt Trigger

It is possible to configure each input for the Schmitt trigger. By default the Schmitt trigger is active. Disabling the Schmitt trigger is requested when using the QTouch® Library.

32.5.14 Parallel Capture Mode

32.5.14.1 Overview

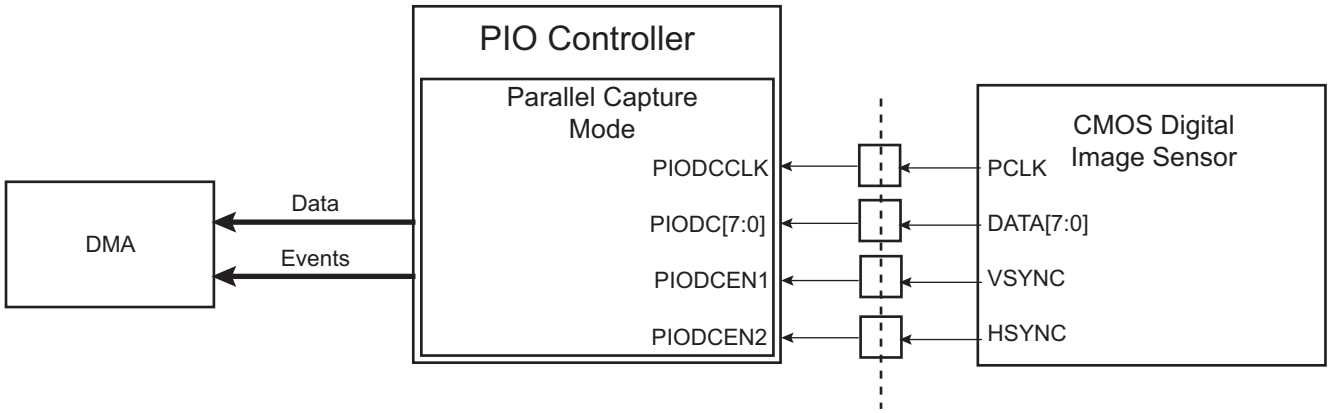
The PIO Controller integrates an interface able to read data from a CMOS digital image sensor, a high-speed parallel ADC, a DSP synchronous port in synchronous mode, etc. For better understanding and to ease reading, the following description uses an example with a CMOS digital image sensor.

32.5.14.2 Functional Description

The CMOS digital image sensor provides a sensor clock, an 8-bit data synchronous with the sensor clock and two data enables which are also synchronous with the sensor clock.

Downloaded from Arrow.com.

Figure 32-8. PIO Controller Connection with CMOS Digital Image Sensor



As soon as the parallel capture mode is enabled by writing a one to the PCEN bit in PIO_PCMR, the I/O lines connected to the sensor clock (PIODCCLK), the sensor data (PIODC[7:0]) and the sensor data enable signals (PIODCEN1 and PIODCEN2) are configured automatically as inputs. To know which I/O lines are associated with the sensor clock, the sensor data and the sensor data enable signals, refer to the I/O multiplexing table(s) in Section 5. “Package and Pinout”.

Once enabled, the parallel capture mode samples the data at rising edge of the sensor clock and resynchronizes it with the peripheral clock domain.

The size of the data which can be read in PIO_PCRHR can be programmed using the DSIZE field in PIO_PCMR. If this data size is larger than 8 bits, then the parallel capture mode samples several sensor data to form a concatenated data of size defined by DSIZE. Then this data is stored in PIO_PCRHR and the flag DRDY is set to one in PIO_PCISR.

The parallel capture mode can be associated with a reception channel of the DMA Controller. This performs reception transfer from parallel capture mode to a memory buffer without any intervention from the CPU.

The parallel capture mode can take into account the sensor data enable signals or not. If the bit ALWYS is set to zero in PIO_PCMR, the parallel capture mode samples the sensor data at the rising edge of the sensor clock only if both data enable signals are active (at one). If the bit ALWYS is set to one, the parallel capture mode samples the sensor data at the rising edge of the sensor clock whichever the data enable signals are.

The parallel capture mode can sample the sensor data only one time out of two. This is particularly useful when the user wants only to sample the luminance Y of a CMOS digital image sensor which outputs a YUV422 data stream. If the HALFS bit is set to zero in PIO_PCMR, the parallel capture mode samples the sensor data in the conditions described above. If the HALFS bit is set to one in PIO_PCMR, the parallel capture mode samples the sensor data in the conditions described above, but only one time out of two. Depending on the FRSTS bit in PIO_PCMR, the sensor can either sample the even or odd sensor data. If sensor data are numbered in the order that they are received with an index from zero to n, if FRSTS equals zero then only data with an even index are sampled. If FRSTS equals one, then only data with an odd index are sampled. If data is ready in PIO_PCRHR and it is not read before a new data is stored in PIO_PCRHR, then an overrun error occurs. The previous data is lost and the OVRE flag in PIO_PCISR is set to one. This flag is automatically reset when PIO_PCISR is read (reset after read).

The flags DRDY and OVRE can be a source of the PIO interrupt.

Downloaded from Arrow.com.

Figure 32-9. Parallel Capture Mode Waveforms (DSIZE = 2, ALWYS = 0, HALFS = 0)

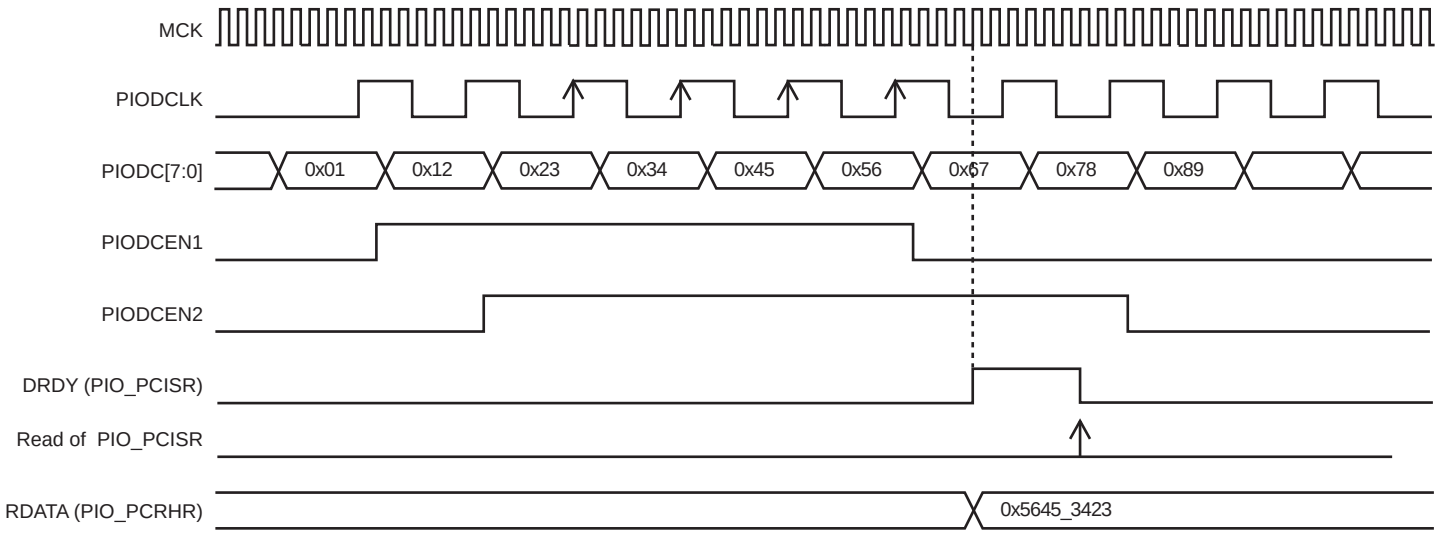
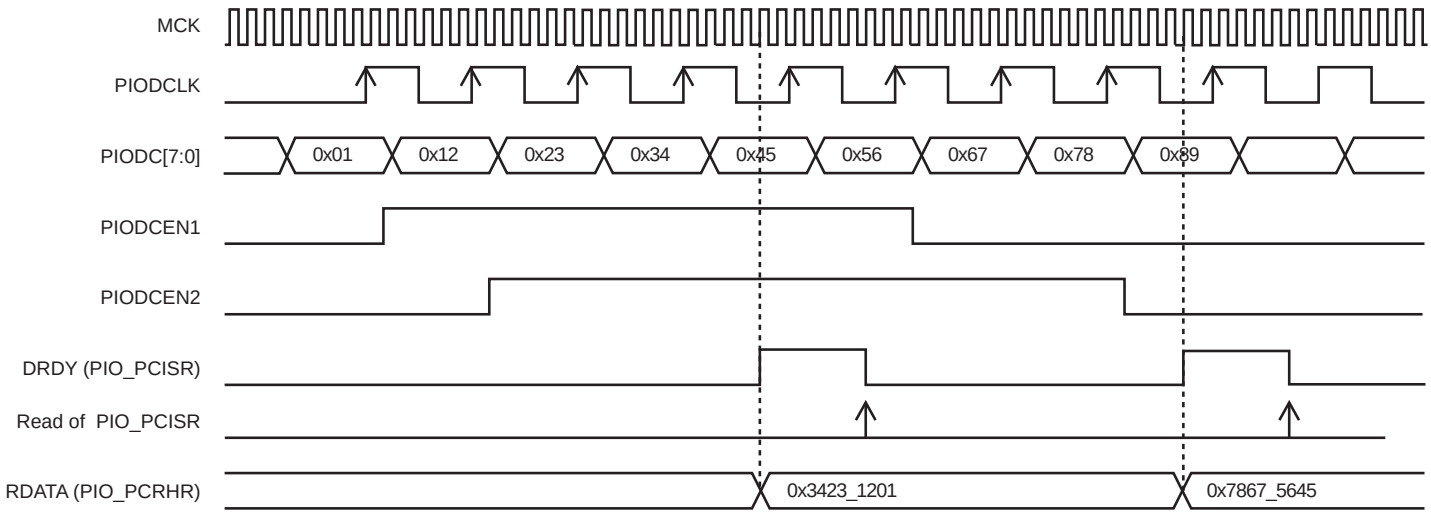


Figure 32-10. Parallel Capture Mode Waveforms (DSIZE = 2, ALWYS = 1, HALFS = 0)



Downloaded from Arrow.com.

Figure 32-11. Parallel Capture Mode Waveforms (DSIZE = 2, ALWYS = 0, HALFS = 1, FRSTS = 0)

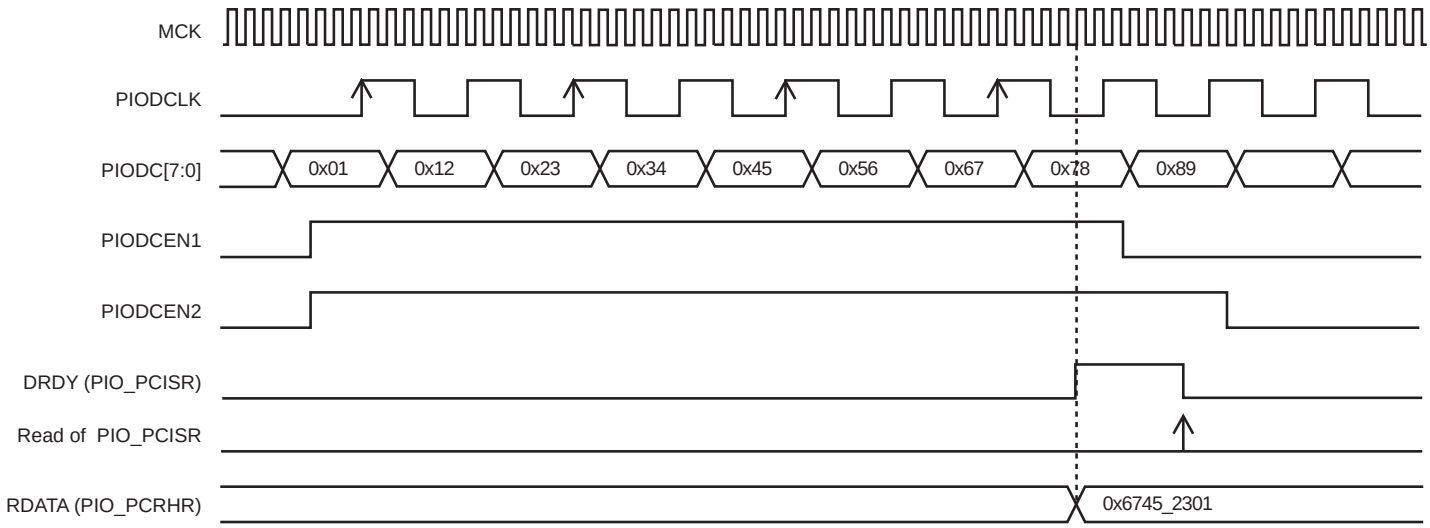
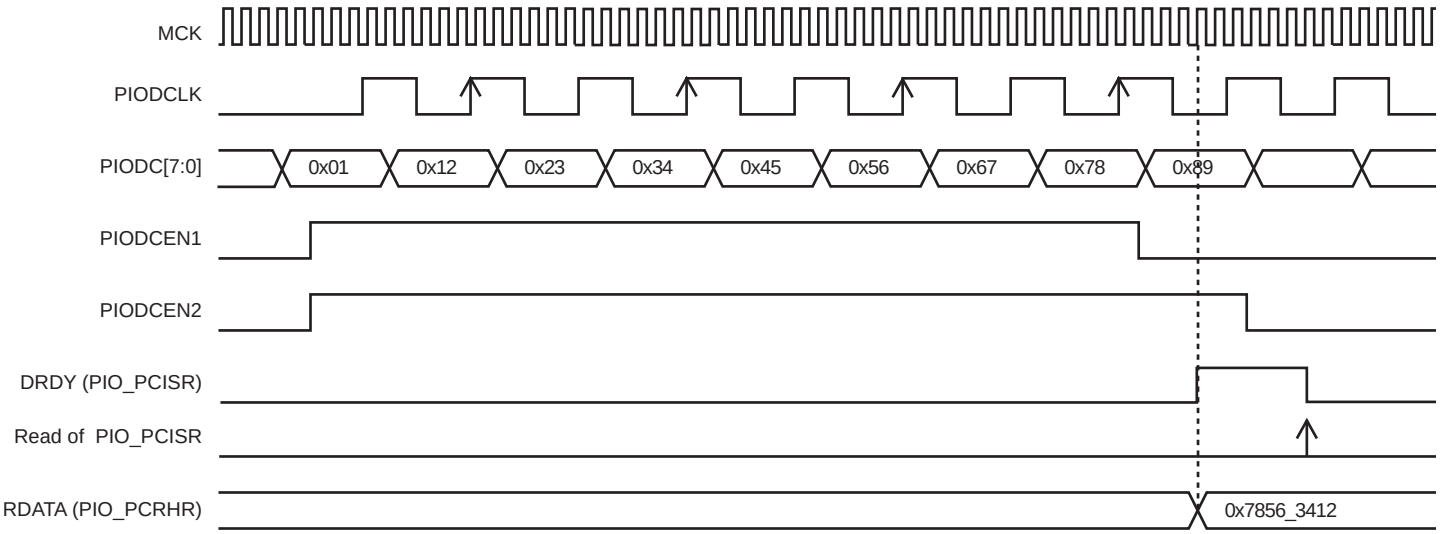


Figure 32-12. Parallel Capture Mode Waveforms (DSIZE = 2, ALWYS = 0, HALFS = 1, FRSTS = 1)



32.5.14.3 Restrictions

- Configuration fields DSIZE, ALWYS, HALFS and FRSTS in PIO_PCMR can be changed **ONLY** if the parallel capture mode is disabled at this time (PCEN = 0 in PIO_PCMR).
- The frequency of peripheral clock must be strictly superior to two times the frequency of the clock of the device which generates the parallel data.

Downloaded from Arrow.com.

32.5.14.4 Programming Sequence

Without DMA

1. Write PIO_PCIDR and PIO_PCIER in order to configure the parallel capture mode interrupt mask.
2. Write PIO_PCMR to set the fields DSIZE, ALWYS, HALFS and FRSTS in order to configure the parallel capture mode **WITHOUT** enabling the parallel capture mode.
3. Write PIO_PCMR to set the PCEN bit to one in order to enable the parallel capture mode **WITHOUT** changing the previous configuration.
4. Wait for a data ready by polling the DRDY flag in PIO_PCISR or by waiting for the corresponding interrupt.
5. Check OVRE flag in PIO_PCISR.
6. Read the data in PIO_PCRHR.
7. If new data are expected, go to step 4.
8. Write PIO_PCMR to set the PCEN bit to zero in order to disable the parallel capture mode **WITHOUT** changing the previous configuration.

With DMA

1. Write PIO_PCIDR and PIO_PCIER in order to configure the parallel capture mode interrupt mask.
2. Configure DMA transfer in DMA registers.
3. Write PIO_PCMR to set the fields DSIZE, ALWYS, HALFS and FRSTS in order to configure the parallel capture mode **WITHOUT** enabling the parallel capture mode.
4. Write PIO_PCMR to set PCEN bit to one in order to enable the parallel capture mode **WITHOUT** changing the previous configuration.
5. Wait for the DMA status flag to indicate that the buffer transfer is complete.
6. Check OVRE flag in PIO_PCISR.
7. If a new buffer transfer is expected, go to step 5.
8. Write PIO_PCMR to set the PCEN bit to zero in order to disable the parallel capture mode **WITHOUT** changing the previous configuration.

Downloaded from Arrow.com.

32.5.15 I/O Lines Programming Example

The programming example shown in Table 32-4 is used to obtain the following configuration:

- 4-bit output port on I/O lines 0 to 3 (should be written in a single write operation), open-drain, with pull-up resistor
- Four output signals on I/O lines 4 to 7 (to drive LEDs for example), driven high and low, no pull-up resistor, no pull-down resistor
- Four input signals on I/O lines 8 to 11 (to read push-button states for example), with pull-up resistors, glitch filters and input change interrupts
- Four input signals on I/O line 12 to 15 to read an external device status (polled, thus no input change interrupt), no pull-up resistor, no glitch filter
- I/O lines 16 to 19 assigned to peripheral A functions with pull-up resistor
- I/O lines 20 to 23 assigned to peripheral B functions with pull-down resistor
- I/O lines 24 to 27 assigned to peripheral C with input change interrupt, no pull-up resistor and no pull-down resistor
- I/O lines 28 to 31 assigned to peripheral D, no pull-up resistor and no pull-down resistor

Table 32-4. Programming Example

Register	Value to be Written
PIO_PER	0x0000_FFFF
PIO_PDR	0xFFFF_0000
PIO_OER	0x0000_00FF
PIO_ODR	0xFFFF_FF00
PIO_IFER	0x0000_0F00
PIO_IFDR	0xFFFF_F0FF
PIO_SODR	0x0000_0000
PIO_CODR	0x0FFF_FFFF
PIO_IER	0x0F00_0F00
PIO_IDR	0xF0FF_F0FF
PIO_MDER	0x0000_000F
PIO_MDDR	0xFFFF_FFF0
PIO_PUDR	0xFFF0_00F0
PIO_PUER	0x000F_FF0F
PIO_PPDDR	0xFF0F_FFFF
PIO_PPDER	0x00F0_0000
PIO_ABCDSR1	0xF0F0_0000
PIO_ABCDSR2	0xFF00_0000
PIO_OWER	0x0000_000F
PIO_OWDR	0x0FFF_ FFF0

Downloaded from Arrow.com.

32.5.16 Register Write Protection

To prevent any single software error from corrupting PIO behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the PIO Write Protection Mode Register (PIO_WPMR).

If a write access to a write-protected register is detected, the WPVS flag in the PIO Write Protection Status Register (PIO_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading the PIO_WPSR.

The following registers can be write-protected:

- PIO Enable Register
- PIO Disable Register
- PIO Output Enable Register
- PIO Output Disable Register
- PIO Input Filter Enable Register
- PIO Input Filter Disable Register
- PIO Multi-driver Enable Register
- PIO Multi-driver Disable Register
- PIO Pull-Up Disable Register
- PIO Pull-Up Enable Register
- PIO Peripheral ABCD Select Register 1
- PIO Peripheral ABCD Select Register 2
- PIO Output Write Enable Register
- PIO Output Write Disable Register
- PIO Pad Pull-Down Disable Register
- PIO Pad Pull-Down Enable Register
- PIO Parallel Capture Mode Register

Downloaded from Arrow.com.

32.6 Parallel Input/Output Controller (PIO) User Interface

Each I/O line controlled by the PIO Controller is associated with a bit in each of the PIO Controller User Interface registers. Each register is 32-bit wide. If a parallel I/O line is not defined, writing to the corresponding bits has no effect. Undefined bits read zero. If the I/O line is not multiplexed with any peripheral, the I/O line is controlled by the PIO Controller and PIO_PSR returns one systematically.

Table 32-5. Register Mapping

Offset	Register	Name	Access	Reset
0x0000	PIO Enable Register	PIO_PER	Write-only	–
0x0004	PIO Disable Register	PIO_PDR	Write-only	–
0x0008	PIO Status Register	PIO_PSR	Read-only	(1)
0x000C	Reserved	–	–	–
0x0010	Output Enable Register	PIO_OER	Write-only	–
0x0014	Output Disable Register	PIO_ODR	Write-only	–
0x0018	Output Status Register	PIO_OSR	Read-only	0x00000000
0x001C	Reserved	–	–	–
0x0020	Glitch Input Filter Enable Register	PIO_IFER	Write-only	–
0x0024	Glitch Input Filter Disable Register	PIO_IFDR	Write-only	–
0x0028	Glitch Input Filter Status Register	PIO_IFSR	Read-only	0x00000000
0x002C	Reserved	–	–	–
0x0030	Set Output Data Register	PIO_SODR	Write-only	–
0x0034	Clear Output Data Register	PIO_CODR	Write-only	
0x0038	Output Data Status Register	PIO_ODSR	Read-only or(2) Read/Write	–
0x003C	Pin Data Status Register	PIO_PDSR	Read-only	(3)
0x0040	Interrupt Enable Register	PIO_IER	Write-only	–
0x0044	Interrupt Disable Register	PIO_IDR	Write-only	–
0x0048	Interrupt Mask Register	PIO_IMR	Read-only	0x00000000
0x004C	Interrupt Status Register(4)	PIO_ISR	Read-only	0x00000000
0x0050	Multi-driver Enable Register	PIO_MDER	Write-only	–
0x0054	Multi-driver Disable Register	PIO_MDDR	Write-only	–
0x0058	Multi-driver Status Register	PIO_MDSR	Read-only	0x00000000
0x005C	Reserved	–	–	–
0x0060	Pull-up Disable Register	PIO_PUDR	Write-only	–
0x0064	Pull-up Enable Register	PIO_PUER	Write-only	–
0x0068	Pad Pull-up Status Register	PIO_PUSR	Read-only	(1)
0x006C	Reserved	–	–	–

Downloaded from Arrow.com.

Table 32-5. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x0070	Peripheral Select Register 1	PIO_ABCDSR1	Read/Write	0x00000000
0x0074	Peripheral Select Register 2	PIO_ABCDSR2	Read/Write	0x00000000
0x0078–0x007C	Reserved	–	–	–
0x0080	Input Filter Slow Clock Disable Register	PIO_IFSCDR	Write-only	–
0x0084	Input Filter Slow Clock Enable Register	PIO_IFSCER	Write-only	–
0x0088	Input Filter Slow Clock Status Register	PIO_IFSCSR	Read-only	0x00000000
0x008C	Slow Clock Divider Debouncing Register	PIO_SCDR	Read/Write	0x00000000
0x0090	Pad Pull-down Disable Register	PIO_PPDDR	Write-only	–
0x0094	Pad Pull-down Enable Register	PIO_PPDER	Write-only	–
0x0098	Pad Pull-down Status Register	PIO_PPDSR	Read-only	(1)
0x009C	Reserved	–	–	–
0x00A0	Output Write Enable	PIO_OWER	Write-only	–
0x00A4	Output Write Disable	PIO_OWDR	Write-only	–
0x00A8	Output Write Status Register	PIO_OWSR	Read-only	0x00000000
0x00AC	Reserved	–	–	–
0x00B0	Additional Interrupt Modes Enable Register	PIO_AIMER	Write-only	–
0x00B4	Additional Interrupt Modes Disable Register	PIO_AIMDR	Write-only	–
0x00B8	Additional Interrupt Modes Mask Register	PIO_AIMMR	Read-only	0x00000000
0x00BC	Reserved	–	–	–
0x00C0	Edge Select Register	PIO_ESR	Write-only	–
0x00C4	Level Select Register	PIO_LSR	Write-only	–
0x00C8	Edge/Level Status Register	PIO_ELSR	Read-only	0x00000000
0x00CC	Reserved	–	–	–
0x00D0	Falling Edge/Low-Level Select Register	PIO_FELLSR	Write-only	–
0x00D4	Rising Edge/High-Level Select Register	PIO_REHLSR	Write-only	–
0x00D8	Fall/Rise - Low/High Status Register	PIO_FRLHSR	Read-only	0x00000000
0x00DC	Reserved	–	–	–
0x00E0	Lock Status	PIO_LOCKSR	Read-only	0x00000000
0x00E4	Write Protection Mode Register	PIO_WPMR	Read/Write	0x00000000
0x00E8	Write Protection Status Register	PIO_WPSR	Read-only	0x00000000
0x00EC–0x00FC	Reserved	–	–	–
0x0100	Schmitt Trigger Register	PIO_SCHMITT	Read/Write	0x00000000
0x0104–0x010C	Reserved	–	–	–
0x0110	Reserved	–	–	–
0x0114	Reserved	–	–	–
0x0118	I/O Drive Register	PIO_DRIVER	Read/Write	0x00000000
0x011C	Reserved	–	–	–

Downloaded from Arrow.com.

Table 32-5. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x0120–0x014C	Reserved	–	–	–
0x0150	Parallel Capture Mode Register	PIO_PCMR	Read/Write	0x00000000
0x0154	Parallel Capture Interrupt Enable Register	PIO_PCIER	Write-only	–
0x0158	Parallel Capture Interrupt Disable Register	PIO_PCIDR	Write-only	–
0x015C	Parallel Capture Interrupt Mask Register	PIO_PCIMR	Read-only	0x00000000
0x0160	Parallel Capture Interrupt Status Register	PIO_PCISR	Read-only	0x00000000
0x0164	Parallel Capture Reception Holding Register	PIO_PCRHR	Read-only	0x00000000
0x0168–0x018C	Reserved	–	–	–

Notes:
1. Reset value depends on the product implementation.
2. PIO_ODSR is Read-only or Read/Write depending on PIO_OWSR I/O lines.
3. Reset value of PIO_PDSR depends on the level of the I/O lines. Reading the I/O line levels requires the clock of the PIO Controller to be enabled, otherwise PIO_PDSR reads the levels present on the I/O line at the time the clock was disabled.
4. PIO_ISR is reset at 0x0. However, the first read of the register may read a different value as input changes may have occurred.
5. If an offset is not listed in the table it must be considered as reserved.

Downloaded from Arrow.com.

32.6.1 PIO Enable Register

Name: PIO_PER

Address: 0x400E0E00 (PIOA), 0x400E1000 (PIOB), 0x400E1200 (PIOC), 0x400E1400 (PIOD), 0x400E1600 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

- **P0–P31: PIO Enable**

0: No effect.

1: Enables the PIO to control the corresponding pin (disables peripheral control of the pin).

Downloaded from Arrow.com.

32.6.2 PIO Disable Register

Name: PIO_PDR

Address: 0x400E0E04 (PIOA), 0x400E1004 (PIOB), 0x400E1204 (PIOC), 0x400E1404 (PIOD), 0x400E1604 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

- **P0–P31: PIO Disable**

0: No effect.

1: Disables the PIO from controlling the corresponding pin (enables peripheral control of the pin).

Downloaded from Arrow.com.

32.6.3 PIO Status Register

Name: PIO_PSR

Address: 0x400E0E08 (PIOA), 0x400E1008 (PIOB), 0x400E1208 (PIOC), 0x400E1408 (PIOD), 0x400E1608 (PIOE)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: PIO Status**

0: PIO is inactive on the corresponding I/O line (peripheral is active).

1: PIO is active on the corresponding I/O line (peripheral is inactive).

Downloaded from Arrow.com.

32.6.4 PIO Output Enable Register

Name: PIO_OER

Address: 0x400E0E10 (PIOA), 0x400E1010 (PIOB), 0x400E1210 (PIOC), 0x400E1410 (PIOD), 0x400E1610 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

- **P0–P31: Output Enable**

0: No effect.

1: Enables the output on the I/O line.

Downloaded from Arrow.com.

32.6.5 PIO Output Disable Register

Name: PIO_ODR

Address: 0x400E0E14 (PIOA), 0x400E1014 (PIOB), 0x400E1214 (PIOC), 0x400E1414 (PIOD), 0x400E1614 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

- **P0–P31: Output Disable**

0: No effect.

1: Disables the output on the I/O line.

Downloaded from Arrow.com.

32.6.6 PIO Output Status Register

Name: PIO_OSR

Address: 0x400E0E18 (PIOA), 0x400E1018 (PIOB), 0x400E1218 (PIOC), 0x400E1418 (PIOD), 0x400E1618 (PIOE)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Output Status**

0: The I/O line is a pure input.

1: The I/O line is enabled in output.

Downloaded from Arrow.com.

32.6.7 PIO Input Filter Enable Register

Name: PIO_IFER

Address: 0x400E0E20 (PIOA), 0x400E1020 (PIOB), 0x400E1220 (PIOC), 0x400E1420 (PIOD), 0x400E1620 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

- **P0–P31: Input Filter Enable**

0: No effect.

1: Enables the input glitch filter on the I/O line.

Downloaded from Arrow.com.

32.6.8 PIO Input Filter Disable Register

Name: PIO_IFDR

Address: 0x400E0E24 (PIOA), 0x400E1024 (PIOB), 0x400E1224 (PIOC), 0x400E1424 (PIOD), 0x400E1624 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

- **P0–P31: Input Filter Disable**

0: No effect.

1: Disables the input glitch filter on the I/O line.

Downloaded from Arrow.com.

32.6.9 PIO Input Filter Status Register

Name: PIO_IFSR

Address: 0x400E0E28 (PIOA), 0x400E1028 (PIOB), 0x400E1228 (PIOC), 0x400E1428 (PIOD), 0x400E1628 (PIOE)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Input Filter Status**

0: The input glitch filter is disabled on the I/O line.

1: The input glitch filter is enabled on the I/O line.

Downloaded from Arrow.com.

32.6.10 PIO Set Output Data Register

Name: PIO_SODR

Address: 0x400E0E30 (PIOA), 0x400E1030 (PIOB), 0x400E1230 (PIOC), 0x400E1430 (PIOD), 0x400E1630 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Set Output Data**

0: No effect.

1: Sets the data to be driven on the I/O line.

Downloaded from Arrow.com.

32.6.11 PIO Clear Output Data Register

Name: PIO_CODR

Address: 0x400E0E34 (PIOA), 0x400E1034 (PIOB), 0x400E1234 (PIOC), 0x400E1434 (PIOD), 0x400E1634 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Clear Output Data**

0: No effect.

1: Clears the data to be driven on the I/O line.

Downloaded from Arrow.com.

32.6.12 PIO Output Data Status Register

Name: PIO_ODSR

Address: 0x400E0E38 (PIOA), 0x400E1038 (PIOB), 0x400E1238 (PIOC), 0x400E1438 (PIOD), 0x400E1638 (PIOE)

Access: Read-only or Read/Write

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Output Data Status**

0: The data to be driven on the I/O line is 0.

1: The data to be driven on the I/O line is 1.

Downloaded from Arrow.com.

32.6.13 PIO Pin Data Status Register

Name: PIO_PDSR

Address: 0x400E0E3C (PIOA), 0x400E103C (PIOB), 0x400E123C (PIOC), 0x400E143C (PIOD), 0x400E163C (PIOE)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Output Data Status**

0: The I/O line is at level 0.

1: The I/O line is at level 1.

Downloaded from Arrow.com.

32.6.14 PIO Interrupt Enable Register

Name: PIO_IER

Address: 0x400E0E40 (PIOA), 0x400E1040 (PIOB), 0x400E1240 (PIOC), 0x400E1440 (PIOD), 0x400E1640 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Input Change Interrupt Enable**

0: No effect.

1: Enables the input change interrupt on the I/O line.

Downloaded from Arrow.com.

32.6.15 PIO Interrupt Disable Register

Name: PIO_IDR

Address: 0x400E0E44 (PIOA), 0x400E1044 (PIOB), 0x400E1244 (PIOC), 0x400E1444 (PIOD), 0x400E1644 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Input Change Interrupt Disable**

0: No effect.

1: Disables the input change interrupt on the I/O line.

Downloaded from Arrow.com.

32.6.16 PIO Interrupt Mask Register

Name: PIO_IMR

Address: 0x400E0E48 (PIOA), 0x400E1048 (PIOB), 0x400E1248 (PIOC), 0x400E1448 (PIOD), 0x400E1648 (PIOE)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Input Change Interrupt Mask**

0: Input change interrupt is disabled on the I/O line.

1: Input change interrupt is enabled on the I/O line.

Downloaded from Arrow.com.

32.6.17 PIO Interrupt Status Register

Name: PIO_ISR

Address: 0x400E0E4C (PIOA), 0x400E104C (PIOB), 0x400E124C (PIOC), 0x400E144C (PIOD), 0x400E164C (PIOE)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Input Change Interrupt Status**

0: No input change has been detected on the I/O line since PIO_ISR was last read or since reset.

1: At least one input change has been detected on the I/O line since PIO_ISR was last read or since reset.

Downloaded from Arrow.com.

32.6.18 PIO Multi-driver Enable Register

Name: PIO_MDER

Address: 0x400E0E50 (PIOA), 0x400E1050 (PIOB), 0x400E1250 (PIOC), 0x400E1450 (PIOD), 0x400E1650 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

- **P0-P31: Multi-drive Enable**

0: No effect.

1: Enables multi-drive on the I/O line.

Downloaded from Arrow.com.

32.6.19 PIO Multi-driver Disable Register

Name: PIO_MDDR

Address: 0x400E0E54 (PIOA), 0x400E1054 (PIOB), 0x400E1254 (PIOC), 0x400E1454 (PIOD), 0x400E1654 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

- **P0–P31: Multi-drive Disable**

0: No effect.

1: Disables multi-drive on the I/O line.

Downloaded from Arrow.com.

32.6.20 PIO Multi-driver Status Register

Name: PIO_MDSR

Address: 0x400E0E58 (PIOA), 0x400E1058 (PIOB), 0x400E1258 (PIOC), 0x400E1458 (PIOD), 0x400E1658 (PIOE)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Multi-drive Status**

0: The multi-drive is disabled on the I/O line. The pin is driven at high- and low-level.

1: The multi-drive is enabled on the I/O line. The pin is driven at low-level only.

Downloaded from Arrow.com.

32.6.21 PIO Pull-Up Disable Register

Name: PIO_PUDR

Address: 0x400E0E60 (PIOA), 0x400E1060 (PIOB), 0x400E1260 (PIOC), 0x400E1460 (PIOD), 0x400E1660 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

- **P0–P31: Pull-Up Disable**

0: No effect.

1: Disables the pull-up resistor on the I/O line.

Downloaded from Arrow.com.

32.6.22 PIO Pull-Up Enable Register

Name: PIO_PUER

Address: 0x400E0E64 (PIOA), 0x400E1064 (PIOB), 0x400E1264 (PIOC), 0x400E1464 (PIOD), 0x400E1664 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

- **P0–P31: Pull-Up Enable**

0: No effect.

1: Enables the pull-up resistor on the I/O line.

Downloaded from Arrow.com.

32.6.23 PIO Pull-Up Status Register

Name: PIO_PUSR

Address: 0x400E0E68 (PIOA), 0x400E1068 (PIOB), 0x400E1268 (PIOC), 0x400E1468 (PIOD), 0x400E1668 (PIOE)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Pull-Up Status**

0: Pull-up resistor is enabled on the I/O line.

1: Pull-up resistor is disabled on the I/O line.

Downloaded from Arrow.com.

32.6.24 PIO Peripheral ABCD Select Register 1

Name: PIO_ABCDSR1

Access: Read/Write

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

- **P0–P31: Peripheral Select**

If the same bit is set to 0 in PIO_ABCDSR2:

0: Assigns the I/O line to the Peripheral A function.

1: Assigns the I/O line to the Peripheral B function.

If the same bit is set to 1 in PIO_ABCDSR2:

0: Assigns the I/O line to the Peripheral C function.

1: Assigns the I/O line to the Peripheral D function.

Downloaded from Arrow.com.

32.6.25 PIO Peripheral ABCD Select Register 2

Name: PIO_ABCDSR2

Address: 0x400E0E70 (PIOA), 0x400E1070 (PIOB), 0x400E1270 (PIOC), 0x400E1470 (PIOD), 0x400E1670 (PIOE)

Access: Read/Write

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

- **P0–P31: Peripheral Select**

If the same bit is set to 0 in PIO_ABCDSR1:

0: Assigns the I/O line to the Peripheral A function.

1: Assigns the I/O line to the Peripheral C function.

If the same bit is set to 1 in PIO_ABCDSR1:

0: Assigns the I/O line to the Peripheral B function.

1: Assigns the I/O line to the Peripheral D function.

Downloaded from Arrow.com.

32.6.26 PIO Input Filter Slow Clock Disable Register

Name: PIO_IFSCDR

Address: 0x400E0E80 (PIOA), 0x400E1080 (PIOB), 0x400E1280 (PIOC), 0x400E1480 (PIOD), 0x400E1680 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Peripheral Clock Glitch Filtering Select**

0: No effect.

1: The glitch filter is able to filter glitches with a duration < $t_{peripheral\ clock}/2$.

Downloaded from Arrow.com.

32.6.27 PIO Input Filter Slow Clock Enable Register

Name: PIO_IFSCER

Address: 0x400E0E84 (PIOA), 0x400E1084 (PIOB), 0x400E1284 (PIOC), 0x400E1484 (PIOD), 0x400E1684 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Slow Clock Debouncing Filtering Select**

0: No effect.

1: The debouncing filter is able to filter pulses with a duration < $t_{div_slck}/2$.

Downloaded from Arrow.com.

32.6.28 PIO Input Filter Slow Clock Status Register

Name: PIO_IFSCSR

Address: 0x400E0E88 (PIOA), 0x400E1088 (PIOB), 0x400E1288 (PIOC), 0x400E1488 (PIOD), 0x400E1688 (PIOE)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Glitch or Debouncing Filter Selection Status**

0: The glitch filter is able to filter glitches with a duration < $t_{peripheral\ clock}/2$.

1: The debouncing filter is able to filter pulses with a duration < $t_{div_slck}/2$.

Downloaded from Arrow.com.

32.6.29 PIO Slow Clock Divider Debouncing Register

Name: PIO_SCDR

Address: 0x400E0E8C (PIOA), 0x400E108C (PIOB), 0x400E128C (PIOC), 0x400E148C (PIOD), 0x400E168C (PIOE)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	DIV					
7	6	5	4	3	2	1	0
DIV							

- **DIV: Slow Clock Divider Selection for Debouncing**

$t_{div_slck} = ((DIV + 1) \times 2) \times t_{slck}$

Downloaded from Arrow.com.

32.6.30 PIO Pad Pull-Down Disable Register

Name: PIO_PPDDR

Address: 0x400E0E90 (PIOA), 0x400E1090 (PIOB), 0x400E1290 (PIOC), 0x400E1490 (PIOD), 0x400E1690 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

- **P0–P31: Pull-Down Disable**

0: No effect.

1: Disables the pull-down resistor on the I/O line.

Downloaded from Arrow.com.

32.6.31 PIO Pad Pull-Down Enable Register

Name: PIO_PPDER

Address: 0x400E0E94 (PIOA), 0x400E1094 (PIOB), 0x400E1294 (PIOC), 0x400E1494 (PIOD), 0x400E1694 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

- **P0–P31: Pull-Down Enable**

0: No effect.

1: Enables the pull-down resistor on the I/O line.

Downloaded from Arrow.com.

32.6.32 PIO Pad Pull-Down Status Register

Name: PIO_PPDSR

Address: 0x400E0E98 (PIOA), 0x400E1098 (PIOB), 0x400E1298 (PIOC), 0x400E1498 (PIOD), 0x400E1698 (PIOE)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Pull-Down Status**

0: Pull-down resistor is enabled on the I/O line.

1: Pull-down resistor is disabled on the I/O line.

Downloaded from Arrow.com.

32.6.33 PIO Output Write Enable Register

Name: PIO_OWER

Address: 0x400E0EA0 (PIOA), 0x400E10A0 (PIOB), 0x400E12A0 (PIOC), 0x400E14A0 (PIOD), 0x400E16A0 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

- **P0–P31: Output Write Enable**

0: No effect.

1: Enables writing PIO_ODSR for the I/O line.

Downloaded from Arrow.com.

32.6.34 PIO Output Write Disable Register

Name: PIO_OWDR

Address: 0x400E0EA4 (PIOA), 0x400E10A4 (PIOB), 0x400E12A4 (PIOC), 0x400E14A4 (PIOD), 0x400E16A4 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

- **P0–P31: Output Write Disable**

0: No effect.

1: Disables writing PIO_ODSR for the I/O line.

Downloaded from Arrow.com.

32.6.35 PIO Output Write Status Register

Name: PIO_OWSR

Address: 0x400E0EA8 (PIOA), 0x400E10A8 (PIOB), 0x400E12A8 (PIOC), 0x400E14A8 (PIOD), 0x400E16A8 (PIOE)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Output Write Status**

0: Writing PIO_ODSR does not affect the I/O line.

1: Writing PIO_ODSR affects the I/O line.

Downloaded from Arrow.com.

32.6.36 PIO Additional Interrupt Modes Enable Register

Name: PIO_AIMER

Address: 0x400E0EB0 (PIOA), 0x400E10B0 (PIOB), 0x400E12B0 (PIOC), 0x400E14B0 (PIOD), 0x400E16B0 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Additional Interrupt Modes Enable**

0: No effect.

1: The interrupt source is the event described in PIO_ELSR and PIO_FRLHSR.

Downloaded from Arrow.com.

32.6.37 PIO Additional Interrupt Modes Disable Register

Name: PIO_AIMDR

Address: 0x400E0EB4 (PIOA), 0x400E10B4 (PIOB), 0x400E12B4 (PIOC), 0x400E14B4 (PIOD), 0x400E16B4 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Additional Interrupt Modes Disable**

0: No effect.

1: The interrupt mode is set to the default interrupt mode (both-edge detection).

Downloaded from Arrow.com.

32.6.38 PIO Additional Interrupt Modes Mask Register

Name: PIO_AIMMR

Address: 0x400E0EB8 (PIOA), 0x400E10B8 (PIOB), 0x400E12B8 (PIOC), 0x400E14B8 (PIOD), 0x400E16B8 (PIOE)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: IO Line Index**

Selects the IO event type triggering an interrupt.

0: The interrupt source is a both-edge detection event.

1: The interrupt source is described by the registers PIO_ELSR and PIO_FRLHSR.

Downloaded from Arrow.com.

32.6.39 PIO Edge Select Register

Name: PIO_ESR

Address: 0x400E0EC0 (PIOA), 0x400E10C0 (PIOB), 0x400E12C0 (PIOC), 0x400E14C0 (PIOD), 0x400E16C0 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Edge Interrupt Selection**

0: No effect.

1: The interrupt source is an edge-detection event.

Downloaded from Arrow.com.

32.6.40 PIO Level Select Register

Name: PIO_LSR

Address: 0x400E0EC4 (PIOA), 0x400E10C4 (PIOB), 0x400E12C4 (PIOC), 0x400E14C4 (PIOD), 0x400E16C4 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Level Interrupt Selection**

0: No effect.

1: The interrupt source is a level-detection event.

Downloaded from Arrow.com.

32.6.41 PIO Edge/Level Status Register

Name: PIO_ELSR

Address: 0x400E0EC8 (PIOA), 0x400E10C8 (PIOB), 0x400E12C8 (PIOC), 0x400E14C8 (PIOD), 0x400E16C8 (PIOE)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Edge/Level Interrupt Source Selection**

0: The interrupt source is an edge-detection event.

1: The interrupt source is a level-detection event.

Downloaded from Arrow.com.

32.6.42 PIO Falling Edge/Low-Level Select Register

Name: PIO_FELLSR

Address: 0x400E0ED0 (PIOA), 0x400E10D0 (PIOB), 0x400E12D0 (PIOC), 0x400E14D0 (PIOD), 0x400E16D0 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Falling Edge/Low-Level Interrupt Selection**

0: No effect.

1: The interrupt source is set to a falling edge detection or low-level detection event, depending on PIO_ELSR.

Downloaded from Arrow.com.

32.6.43 PIO Rising Edge/High-Level Select Register

Name: PIO_REHLSR

Address: 0x400E0ED4 (PIOA), 0x400E10D4 (PIOB), 0x400E12D4 (PIOC), 0x400E14D4 (PIOD), 0x400E16D4 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Rising Edge/High-Level Interrupt Selection**

0: No effect.

1: The interrupt source is set to a rising edge detection or high-level detection event, depending on PIO_ELSR.

Downloaded from Arrow.com.

32.6.44 PIO Fall/Rise - Low/High Status Register

Name: PIO_FRLHSR

Address: 0x400E0ED8 (PIOA), 0x400E10D8 (PIOB), 0x400E12D8 (PIOC), 0x400E14D8 (PIOD), 0x400E16D8 (PIOE)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Edge/Level Interrupt Source Selection**

0: The interrupt source is a falling edge detection (if PIO_ELSR = 0) or low-level detection event (if PIO_ELSR = 1).

1: The interrupt source is a rising edge detection (if PIO_ELSR = 0) or high-level detection event (if PIO_ELSR = 1).

Downloaded from Arrow.com.

32.6.45 PIO Lock Status Register

Name: PIO_LOCKSR

Address: 0x400E0EE0 (PIOA), 0x400E10E0 (PIOB), 0x400E12E0 (PIOC), 0x400E14E0 (PIOD), 0x400E16E0 (PIOE)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Lock Status**

0: The I/O line is not locked.

1: The I/O line is locked.

Downloaded from Arrow.com.

32.6.46 PIO Write Protection Mode Register

Name: PIO_WPMR

Address: 0x400E0EE4 (PIOA), 0x400E10E4 (PIOB), 0x400E12E4 (PIOC), 0x400E14E4 (PIOD), 0x400E16E4 (PIOE)

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							
23	22	21	20	19	18	17	16
WPKEY							
15	14	13	12	11	10	9	8
WPKEY							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x50494F (“PIO” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x50494F (“PIO” in ASCII).

See Section 32.5.16 “Register Write Protection” for the list of registers that can be protected.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x50494F	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

Downloaded from Arrow.com.

32.6.47 PIO Write Protection Status Register

Name: PIO_WPSR

Address: 0x400E0EE8 (PIOA), 0x400E10E8 (PIOB), 0x400E12E8 (PIOC), 0x400E14E8 (PIOD), 0x400E16E8 (PIOE)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
WPVSRC							
15	14	13	12	11	10	9	8
WPVSRC							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

- **WPVS: Write Protection Violation Status**

0: No write protection violation has occurred since the last read of the PIO_WPSR.

1: A write protection violation has occurred since the last read of the PIO_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protection Violation Source**

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

Downloaded from Arrow.com.

32.6.48 PIO Schmitt Trigger Register

Name: PIO_SCHMITT

Address: 0x400E0F00 (PIOA), 0x400E1100 (PIOB), 0x400E1300 (PIOC), 0x400E1500 (PIOD), 0x400E1700 (PIOE)

Access: Read/Write

31	30	29	28	27	26	25	24
SCHMITT31	SCHMITT30	SCHMITT29	SCHMITT28	SCHMITT27	SCHMITT26	SCHMITT25	SCHMITT24
23	22	21	20	19	18	17	16
SCHMITT23	SCHMITT22	SCHMITT21	SCHMITT20	SCHMITT19	SCHMITT18	SCHMITT17	SCHMITT16
15	14	13	12	11	10	9	8
SCHMITT15	SCHMITT14	SCHMITT13	SCHMITT12	SCHMITT11	SCHMITT10	SCHMITT9	SCHMITT8
7	6	5	4	3	2	1	0
SCHMITT7	SCHMITT6	SCHMITT5	SCHMITT4	SCHMITT3	SCHMITT2	SCHMITT1	SCHMITT0

- **SCHMITTx [x=0..31]: Schmitt Trigger Control**

0: Schmitt trigger is enabled.

1: Schmitt trigger is disabled.

Downloaded from Arrow.com.

32.6.49 PIO I/O Drive Register

Name: PIO_DRIVER

Address: 0x400E0F18 (PIOA), 0x400E1118 (PIOB), 0x400E1318 (PIOC), 0x400E1518 (PIOD), 0x400E1718 (PIOE)

Access: Read/Write

31	30	29	28	27	26	25	24
LINE31	LINE30	LINE29	LINE28	LINE27	LINE26	LINE25	LINE24
23	22	21	20	19	18	17	16
LINE23	LINE22	LINE21	LINE20	LINE19	LINE18	LINE17	LINE16
15	14	13	12	11	10	9	8
LINE15	LINE14	LINE13	LINE12	LINE11	LINE10	LINE9	LINE8
7	6	5	4	3	2	1	0
LINE7	LINE6	LINE5	LINE4	LINE3	LINE2	LINE1	LINE0

- **LINEx [x=0..31]: Drive of PIO Line x**

Value	Name	Description
0	LOW_DRIVE	Lowest drive
1	HIGH_DRIVE	Highest drive

Downloaded from Arrow.com.

32.6.50 PIO Parallel Capture Mode Register

Name: PIO_PCMR

Address: 0x400E0F50 (PIOA), 0x400E1150 (PIOB), 0x400E1350 (PIOC), 0x400E1550 (PIOD), 0x400E1750 (PIOE)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	FRSTS	HALFS	ALWYS	–
7	6	5	4	3	2	1	0
–	–	DSIZE		–	–	–	PCEN

This register can only be written if the WPEN bit is cleared in the PIO Write Protection Mode Register.

- **PCEN: Parallel Capture Mode Enable**

0: The parallel capture mode is disabled.

1: The parallel capture mode is enabled.

- **DSIZE: Parallel Capture Mode Data Size**

Value	Name	Description
0	BYTE	The reception data in the PIO_PCRHR is a byte (8-bit)
1	HALF-WORD	The reception data in the PIO_PCRHR is a half-word (16-bit)
2	WORD	The reception data in the PIO_PCRHR is a word (32-bit)
3	–	Reserved

- **ALWYS: Parallel Capture Mode Always Sampling**

0: The parallel capture mode samples the data when both data enables are active.

1: The parallel capture mode samples the data whatever the data enables are.

- **HALFS: Parallel Capture Mode Half Sampling**

Independently from the ALWYS bit:

0: The parallel capture mode samples all the data.

1: The parallel capture mode samples the data only every other time.

- **FRSTS: Parallel Capture Mode First Sample**

This bit is useful only if the HALFS bit is set to 1. If data are numbered in the order that they are received with an index from 0 to n:

0: Only data with an even index are sampled.

1: Only data with an odd index are sampled.

Downloaded from Arrow.com.

32.6.51 PIO Parallel Capture Interrupt Enable Register

Name: PIO_PCIER

Address: 0x400E0F54 (PIOA), 0x400E1154 (PIOB), 0x400E1354 (PIOC), 0x400E1554 (PIOD), 0x400E1754 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	RXBUFF	ENDRX	OVRE	DRDY

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Enables the corresponding interrupt

- **DRDY: Parallel Capture Mode Data Ready Interrupt Enable**
- **OVRE: Parallel Capture Mode Overrun Error Interrupt Enable**
- **ENDRX: End of Reception Transfer Interrupt Enable**
- **RXBUFF: Reception Buffer Full Interrupt Enable**

Downloaded from Arrow.com.

32.6.52 PIO Parallel Capture Interrupt Disable Register

Name: PIO_PCIDR

Address: 0x400E0F58 (PIOA), 0x400E1158 (PIOB), 0x400E1358 (PIOC), 0x400E1558 (PIOD), 0x400E1758 (PIOE)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	RXBUFF	ENDRX	OVRE	DRDY

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Disables the corresponding interrupt

- **DRDY: Parallel Capture Mode Data Ready Interrupt Disable**
- **OVRE: Parallel Capture Mode Overrun Error Interrupt Disable**
- **ENDRX: End of Reception Transfer Interrupt Disable**
- **RXBUFF: Reception Buffer Full Interrupt Disable**

Downloaded from Arrow.com.

32.6.53 PIO Parallel Capture Interrupt Mask Register

Name: PIO_PCIMR

Address: 0x400E0F5C (PIOA), 0x400E115C (PIOB), 0x400E135C (PIOC), 0x400E155C (PIOD), 0x400E175C (PIOE)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	RXBUFF	ENDRX	OVRE	DRDY

The following configuration values are valid for all listed bit names of this register:

0: Corresponding interrupt is not enabled.

1: Corresponding interrupt is enabled.

- **DRDY: Parallel Capture Mode Data Ready Interrupt Mask**
- **OVRE: Parallel Capture Mode Overrun Error Interrupt Mask**
- **ENDRX: End of Reception Transfer Interrupt Mask**
- **RXBUFF: Reception Buffer Full Interrupt Mask**

Downloaded from Arrow.com.

32.6.54 PIO Parallel Capture Interrupt Status Register

Name: PIO_PCISR

Address: 0x400E0F60 (PIOA), 0x400E1160 (PIOB), 0x400E1360 (PIOC), 0x400E1560 (PIOD), 0x400E1760 (PIOE)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	OVRE	DRDY

- **DRDY: Parallel Capture Mode Data Ready**

0: No new data is ready to be read since the last read of PIO_PCRHR.

1: A new data is ready to be read since the last read of PIO_PCRHR.

The DRDY flag is automatically reset when PIO_PCRHR is read or when the parallel capture mode is disabled.

- **OVRE: Parallel Capture Mode Overrun Error**

0: No overrun error occurred since the last read of this register.

1: At least one overrun error occurred since the last read of this register.

The OVRE flag is automatically reset when this register is read or when the parallel capture mode is disabled.

Downloaded from Arrow.com.

32.6.55 PIO Parallel Capture Reception Holding Register

Name: PIO_PCRHR

Address: 0x400E0F64 (PIOA), 0x400E1164 (PIOB), 0x400E1364 (PIOC), 0x400E1564 (PIOD), 0x400E1764 (PIOE)

Access: Read-only

31	30	29	28	27	26	25	24
RDATA							
23	22	21	20	19	18	17	16
RDATA							
15	14	13	12	11	10	9	8
RDATA							
7	6	5	4	3	2	1	0
RDATA							

- **RDATA: Parallel Capture Mode Reception Data**

If DSIZE = 0 in PIO_PCMR, only the 8 LSBs of RDATA are useful.

If DSIZE = 1 in PIO_PCMR, only the 16 LSBs of RDATA are useful.

Downloaded from Arrow.com.

33. External Bus Interface (EBI)

33.1 Description

The External Bus Interface (EBI) is designed to ensure the successful data transfer between several external devices and the embedded Memory Controller of an ARM-based device.

The Static Memory and SDRAM Controllers are all featured external Memory Controllers on the EBI. These external Memory Controllers are capable of handling several types of external memory and peripheral devices, such as SRAM, PROM, EPROM, EEPROM, Flash and SDR-SDRAM. The EBI operates with 1.8V or 3.3V Power Supply (VDDIO).

The EBI also supports the NAND Flash protocols via integrated circuitry that greatly reduces the requirements for external components. Furthermore, the EBI handles data transfers with up to six external devices, each assigned to six address spaces defined by the embedded Memory Controller. Data transfers are performed through a 16-bit or 32-bit data bus, an address bus of up to 24 bits, up to four chip select lines (NCS[3:0]) and several control pins that are generally multiplexed between the different external Memory Controllers.

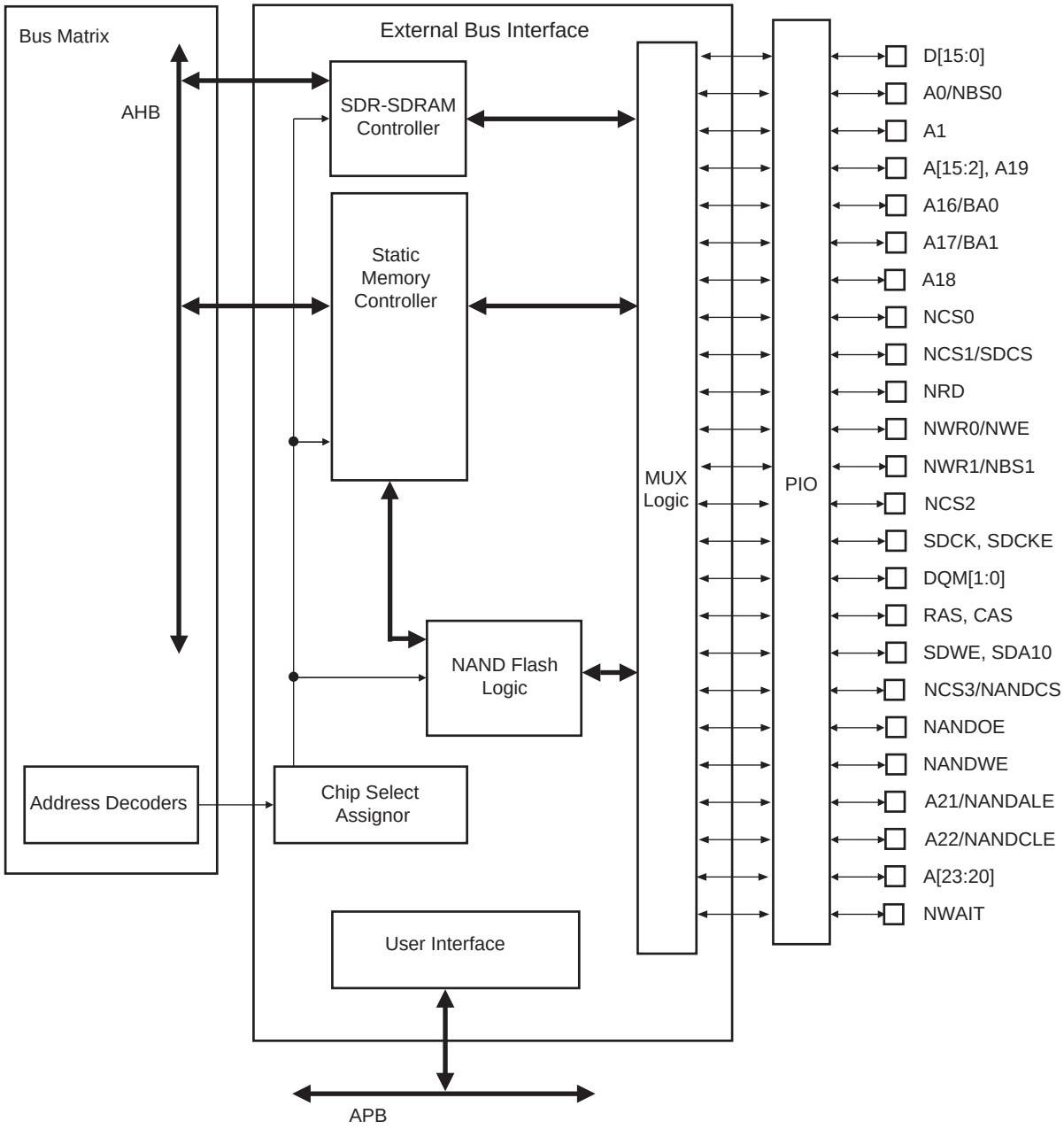
33.2 Embedded Characteristics

- Integrates two External Memory Controllers
  - Static Memory Controller
  - SDR-SDRAM Controller
- Integrates NAND Flash Logic
- Up to 24-bit Address Bus (up to 16 Mbytes linear per chip select)
- Up to four Chip Selects, Configurable Assignment
  - Static Memory Controller on NCS0, NCS1, NCS2, NCS3
  - SDR-SDRAM Controller (SDCS) or Static Memory Controller on NCS1
  - NAND Flash support on NCS0, NCS1, NSCS2 and NCS3

Downloaded from Arrow.com.

33.3 EBI Block Diagram

Figure 33-1. Organization of the External Bus Interface



Downloaded from Arrow.com.

33.4 I/O Lines Description

Table 33-1. EBI I/O Lines Description

Name	Function	Type	Active Level
EBI			
D0–D15	Data Bus	I/O	
A0–A23	Address Bus	Output	
NWAIT	External Wait Signal	Input	Low
SMC			
NCS0–EBI_NCS3	Chip Select Lines	Output	Low
NWR0–NWR1	Write Signals	Output	Low
NRD	Read Signal	Output	Low
NWE	Write Enable	Output	Low
NBS0–NBS1	Byte Mask Signals	Output	Low
EBI for NAND Flash Support			
NANDCS	NAND Flash Chip Select Line	Output	Low
NANDOE	NAND Flash Output Enable	Output	Low
NANDWE	NAND Flash Write Enable	Output	Low
SDRAM Controller			
SDCK	SDR-SDRAM Clock	Output	
SDCKE	SDR-SDRAM Clock Enable	Output	High
SDCS	SDR-SDRAM Controller Chip Select Line	Output	Low
BA0–1	Bank Select	Output	
SDWE	SDR-SDRAM Write Enable	Output	Low
RAS - CAS	Row and Column Signal	Output	Low
SDA10	SDRAM Address 10 Line	Output	

The connection of some signals through the MUX logic is not direct and depends on the Memory Controller in use at the moment.

Table 33-2 details the connections between the two Memory Controllers and the EBI pins.

Table 33-2. EBI Pins and Memory Controllers I/O Lines Connections

EBIx Pins	SDRAM I/O Lines	SMC I/O Lines
NWR1/NBS1	NBS1	NWR1
A0/NBS0	Not Supported	SMC_A0
A1	Not Supported	SMC_A1
A[11:2]	SDRAMC_A[9:0]	SMC_A[11:2]
SDA10	SDRAMC_A10	Not Supported
A12	Not Supported	SMC_A12
A[15:13]	SDRAMC_A[13:11]	SMC_A[15:13]
A[25:16]	Not Supported	SMC_A[25:16]
D[15:0]	D[15:0]	D[15:0]

Downloaded from Arrow.com.

33.5 Application Example

33.5.1 Hardware Interface

Table 33-3 details the connections to be applied between the EBI pins and the external devices for each Memory Controller.

Table 33-3. EBI Pins and External Static Device Connections

Signals: EBI_	Pins of the Interfaced Device		
	8-bit Static Device	2 x 8-bit Static Devices	16-bit Static Device
Controller	**SMC**		
D0–D7	D0–D7	D0–D7	D0–D7
D8–D15	–	D8–D15	D8–D15
A0/NBS0	A0	–	NLB
A1	A1	A0	A0
A2–A23	A[2:23]	A[1:22]	A[1:22]
NCS0	CS	CS	CS
NCS1/DDRSDCS	CS	CS	CS
NCS2	CS	CS	CS
NCS3/NANDCS	CS	CS	CS
NRD	OE	OE	OE
NWR0/NWE	WE	WE (1)	WE
NWR1/NBS1	–	WE (1)	NUB

Notes: 1. NWR1 enables upper byte writes. NWR0 enables lower byte writes.

Downloaded from Arrow.com.

Table 33-4. EBI Pins and External Device Connections

Signals: EBI_		Pins of the Interfaced Device	
		SDR/LPSDR	NAND Flash
Controller	Power supply	SDRAMC	NFC
D0–D15	VDDIO	D0–D15	D0–D15
A0/NBS0	VDDIO	DQM0	–
A1	VDDIO	–	–
A2–A10	VDDIO	A[0:8]	–
A11	VDDIO	A9	–
SDA10	VDDIO	A10	–
A12	VDDIO	–	–
A13–A14	VDDIO	A[11:12]	–
A15	VDDIO	A13	–
A16/BA0	VDDIO	BA0	–
A17/BA1	VDDIO	BA1	–
A18	VDDIO	–	–
A19	VDDIO	–	–
A20	VDDIO	–	–
A21/NANDALE	VDDIO	–	ALE
A22/NANDCLE	VDDIO	–	CLE
A23	VDDIO	–	–
NCS0	VDDIO	–	–
NCS1/SDCS	VDDIO	SDCS	–
NCS2	VDDIO	–	–
NCS3/NANDCS	VDDIO	–	CE
NANDOE	VDDIO	–	OE
NANDWE	VDDIO	–	WE
NRD	VDDIO	–	–
NWR0/NWE	VDDIO	–	–
NWR1/NBS1	VDDIO	DQM1	–
SDCK	VDDIO	CK	–
SDCKE	VDDIO	CKE	–
RAS	VDDIO	RAS	–
CAS	VDDIO	CAS	–
SDWE	VDDIO	WE	–
Pxx	VDDIO	–	CE
Pxx	VDDIO	–	RDY

Downloaded from Arrow.com.

33.5.2 Product Dependencies

33.5.2.1 I/O Lines

The pins used for interfacing the External Bus Interface may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the External Bus Interface pins to their peripheral function. If I/O lines of the External Bus Interface are not used by the application, they can be used for other purposes by the PIO Controller.

33.5.3 Functional Description

The EBI transfers data between the internal AHB Bus (handled by the Bus Matrix) and the external memories or peripheral devices. It controls the waveforms and the parameters of the external address, data and control buses and is composed of the following elements:

- Static Memory Controller (SMC)
- SDR-SDRAM Controller (SDRC)
- A chip select assignment feature that assigns an AHB address space to the external devices
- A multiplex controller circuit that shares the pins between the different Memory Controllers
- Programmable NAND Flash support logic

33.5.3.1 Bus Multiplexing

The EBI offers a complete set of control signals that share the 16-bit data lines, the address lines of up to 24 bits and the control signals through a multiplex logic operating in function of the memory area requests.

Multiplexing is specifically organized in order to guarantee the maintenance of the address and output control lines at a stable state while no external access is being performed. Multiplexing is also designed to respect the data float times defined in the Memory Controllers. Furthermore, refresh cycles of the SDR-SDRAM are executed independently by the SDR Controller without delaying the other external Memory Controller accesses.

33.5.3.2 Static Memory Controller

For information on the Static Memory Controller, refer to Section 34. “Static Memory Controller (SMC)”

33.5.3.3 SDRAM Controller

For information on the SDR Controller, refer to Section 28. “SDRAM Controller (SDRAMC)”.

33.5.3.4 NAND Flash Support

External Bus Interfaces integrate circuitry that interfaces to NAND Flash devices.

To ensure that the processor preserves transaction order and thus the correct NAND Flash behavior, the NAND Flash address space is to be declared in the Memory Protection Unit (MPU) as "Device" or "Strongly-ordered" memory. Refer to the *ARM Cortex-M7 Technical Reference Manual* (ARM DDI 0489) available on www.arm.com.

External Bus Interface

The NAND Flash Chip Select (NANDCS) is driven by the Static Memory Controller on the NCS0, NCS1, NCS2 or NCS3 address space depending on value of SMC_SMCSx bits. For example, programming the SMC_NFC3 field in the CCFG_SMCNFCS Register in the Chip Configuration User Interface to the appropriate value enables the NAND Flash logic. For details on this register, refer to Section 18. “Bus Matrix (MATRIX)”. Access to an external NAND Flash device is then made by accessing the address space reserved to NCS3 (i.e., between 0x6300 0000 and 0x6FFF FFFF).

The NAND Flash logic drives the read and write command signals of the SMC on the NANDOE and NANDWE signals when the required SMC_NFCSx signal is active. NANDOE and NANDWE are invalidated as soon as the transfer address fails to lie in the selected NCSx address space. For details on these waveforms, refer to Section 34. “Static Memory Controller (SMC)”.

Downloaded from Arrow.com.

NAND Flash Signals

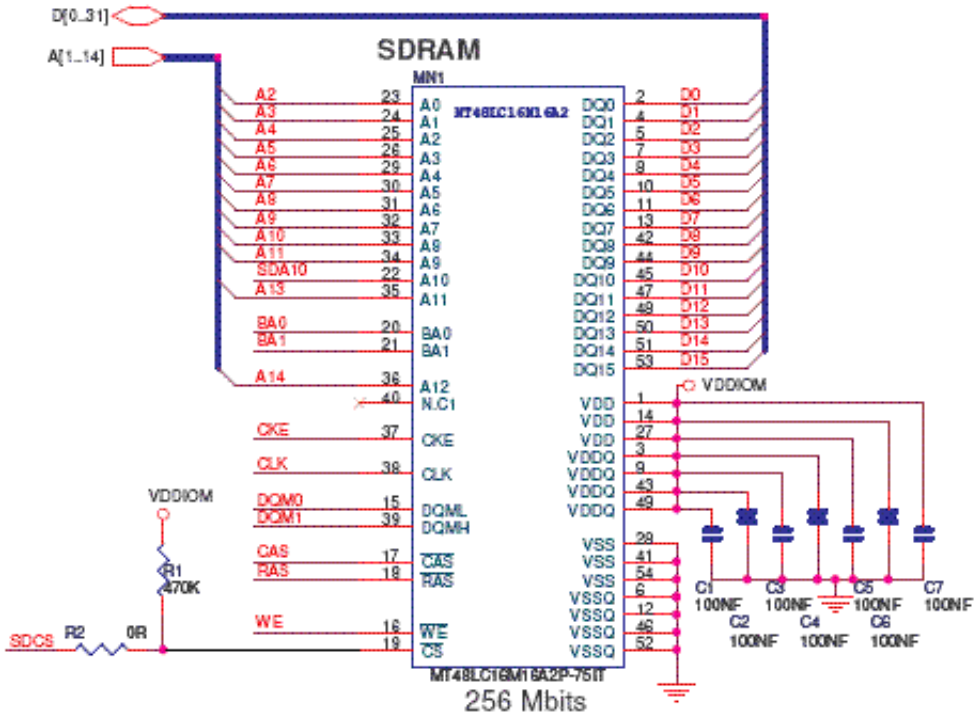
The address latch enable and command latch enable signals on the NAND Flash device are driven by address bits A22 and A21 of the EBI address bus. The command, address or data words on the data bus of the NAND Flash device are distinguished by using their address within the NCSx address space. The chip enable (CE) signal of the device and the ready/busy (R/B) signals are connected to PIO lines. The CE signal then remains asserted even when NCSx is not selected, preventing the device from returning to standby mode.

33.5.4 Implementation Examples

The following hardware configurations are given for illustration only. The user should refer to the memory manufacturer web site to check current device availability.

33.5.4.1 16-bit SDRAM on NCS1

Hardware Configuration



Software Configuration

The following configuration has to be performed:

- Enable the SDRAM support by setting the bit SDRAMEN field in the CCFG_SMCNFCS Register in the Bus Matrix.
- Initialize the SDRAM Controller depending on the SDRAM device and system bus frequency.

The Data Bus Width is to be programmed to 16 bits.

The SDRAM initialization sequence is described in Section 28.5.1 “SDRAM Device Initialization”.

Downloaded from Arrow.com.

34. Static Memory Controller (SMC)

34.1 Description

The External Bus Interface (EBI) is designed to ensure the successful data transfer between several external devices and the ARM-based microcontroller. The Static Memory Controller (SMC) is part of the EBI.

The SMC handles several types of external memory and peripheral devices, such as SRAM, PSRAM, PROM, EPROM, EEPROM, LCD Module, NOR Flash and NAND Flash.

The SMC generates the signals that control the access to the external memory devices or peripheral devices. It has 4 chip selects, a 24-bit address bus, and a configurable 8 or 16-bit data bus. Separate read and write control signals allow for direct memory and peripheral interfacing. Read and write signal waveforms are fully adjustable.

The SMC can manage wait requests from external devices to extend the current access. The SMC is provided with an automatic Slow clock mode. In Slow clock mode, it switches from user-programmed waveforms to slow-rate specific waveforms on read and write signals. The SMC supports asynchronous burst read in Page mode access for page sizes up to 32 bytes.

The external data bus can be scrambled/unscrambled by means of user keys.

34.2 Embedded Characteristics

- Four Chip Selects Available
- 16-Mbyte Address Space per Chip Select
- 8-bit or 16-bit Data Bus
- Zero Wait State Scrambling/Unscrambling Function with User Key
- Word, Halfword, Byte Transfers
- Byte Write or Byte Select Lines
- Programmable Setup, Pulse And Hold Time for Read Signals per Chip Select
- Programmable Setup, Pulse And Hold Time for Write Signals per Chip Select
- Programmable Data Float Time per Chip Select
- External Wait Request
- Automatic Switch to Slow Clock Mode
- Asynchronous Read in Page Mode Supported: Page Size Ranges from 4 to 32 Bytes
- Register Write Protection

Downloaded from Arrow.com.

34.3 I/O Lines Description

Table 34-1. I/O Line Description

Name	Description	Type	Active Level
NCS[3:0]	Static Memory Controller Chip Select Lines	Output	Low
NRD	Read Signal	Output	Low
NWR0/NWE	Write 0/Write Enable Signal	Output	Low
NWR1/NBS1	Write 1/Byte 1 Select Signal	Output	Low
A0/NBS0	Address Bit 0/Byte 0 Select Signal	Output	Low
A[23:1]	Address Bus	Output	–
D[15:0]	Data Bus	I/O	–
NWAIT	External Wait Signal	Input	Low
NANDCS	NAND Flash Chip Select Line	Output	Low
NANDOE	NAND Flash Output Enable	Output	Low
NANDWE	NAND Flash Write Enable	Output	Low
NANDALE	NAND Flash Address Latch Enable	Output	–
NANDCLE	NAND Flash Command Latch Enable	Output	–

34.4 Multiplexed Signals

Table 34-2. Static Memory Controller (SMC) Multiplexed Signals

Multiplexed Signals		Related Function
NWR0	NWE	Byte-write or Byte-select access. See Section 34.7.2.1 ”Byte Write Access” and Section 34.7.2.2 ”Byte Select Access”
A0	NBS0	8-bit or 16-bit data bus. See Section 34.7.1 ”Data Bus Width”
NWR1	NBS1	Byte-write or Byte-select access. See Section 34.7.2.1 ”Byte Write Access” and Section 34.7.2.2 ”Byte Select Access”
A22	NANDCLE	NAND Flash Command Latch Enable
A21	NANDALE	NAND Flash Address Latch Enable

Downloaded from Arrow.com.

34.5 Product Dependencies

34.5.1 I/O Lines

The pins used for interfacing the SMC are multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the SMC pins to their peripheral function. If I/O lines of the SMC are not used by the application, they can be used for other purposes by the PIO Controller.

Table 34-3. I/O Lines

Instance	Signal	I/O Line	Peripheral
SMC	A0/NBS0	PC18	A
SMC	A1	PC19	A
SMC	A2	PC20	A
SMC	A3	PC21	A
SMC	A4	PC22	A
SMC	A5	PC23	A
SMC	A6	PC24	A
SMC	A7	PC25	A
SMC	A8	PC26	A
SMC	A9	PC27	A
SMC	A10	PC28	A
SMC	A11	PC29	A
SMC	A12	PC30	A
SMC	A13	PC31	A
SMC	A14	PA18	C
SMC	A15	PA19	C
SMC	A16/BA0	PA20	C
SMC	A17/BA1	PA0	C
SMC	A18	PA1	C
SMC	A19	PA23	C
SMC	A20	PA24	C
SMC	A21/NANDALE	PC16	A
SMC	A22/NANDCLE	PC17	A
SMC	A23	PA25	C
SMC	CAS	PD17	C
SMC	D0	PC0	A
SMC	D1	PC1	A
SMC	D2	PC2	A
SMC	D3	PC3	A
SMC	D4	PC4	A
SMC	D5	PC5	A

Downloaded from Arrow.com.

Table 34-3. I/O Lines

SMC	D6	PC6	A
SMC	D7	PC7	A
SMC	D8	PE0	A
SMC	D9	PE1	A
SMC	D10	PE2	A
SMC	D11	PE3	A
SMC	D12	PE4	A
SMC	D13	PE5	A
SMC	D14	PA15	A
SMC	D15	PA16	A
SMC	NANDOE	PC9	A
SMC	NANDWE	PC10	A
SMC	NCS0	PC14	A
SMC	NCS1/SDCS	PC15	A
SMC	NCS1/SDCS	PD18	A
SMC	NCS2	PA22	C
SMC	NCS3	PC12	A
SMC	NCS3	PD19	A
SMC	NRD	PC11	A
SMC	NWAIT	PC13	A
SMC	NWR0/NWE	PC8	A
SMC	NWR1/NBS1	PD15	C
SMC	RAS	PD16	C
SMC	SDA10	PC13	C
SMC	SDA10	PD13	C
SMC	SDCK	PD23	C
SMC	SDCKE	PD14	C
SMC	SDWE	PD29	C

34.5.2 Power Management

The SMC is clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the SMC clock.

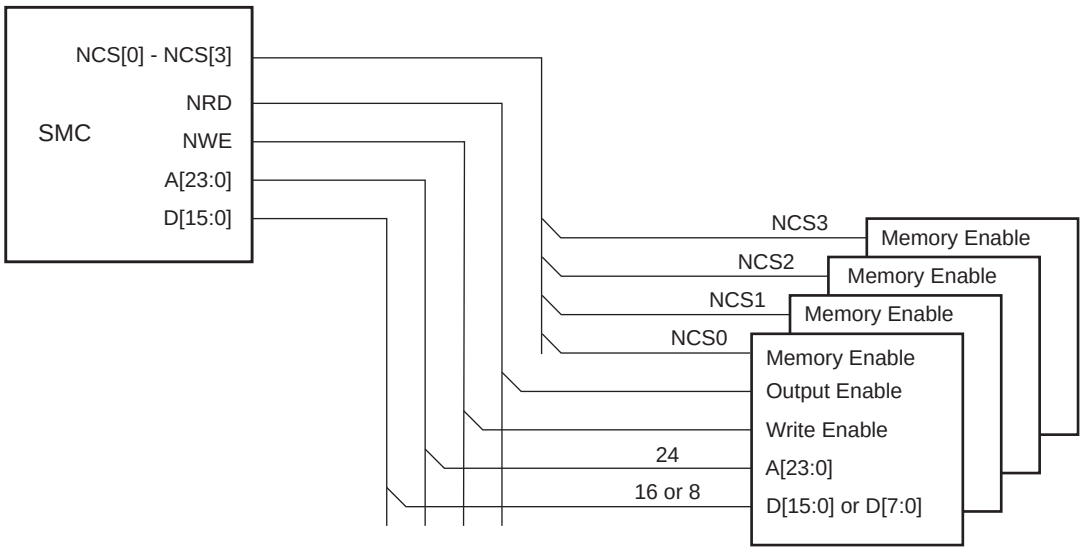
Downloaded from Arrow.com.

34.6 External Memory Mapping

The SMC provides up to 24 address lines, A[23:0]. This allows each chip select line to address up to 16 Mbytes of memory.

If the physical memory device connected on one chip select is smaller than 16 Mbytes, it wraps around and appears to be repeated within this space. The SMC correctly handles any valid access to the memory device within the page (see Figure 34-1).

Figure 34-1. Memory Connections for Four External Devices



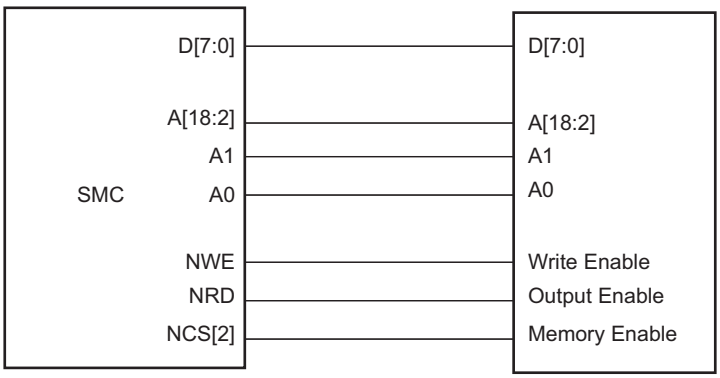
34.7 Connection to External Devices

34.7.1 Data Bus Width

A data bus width of 8 or 16 bits can be selected for each chip select. This option is controlled by the bit DBW in the Mode register (SMC_MODE) for the corresponding chip select.

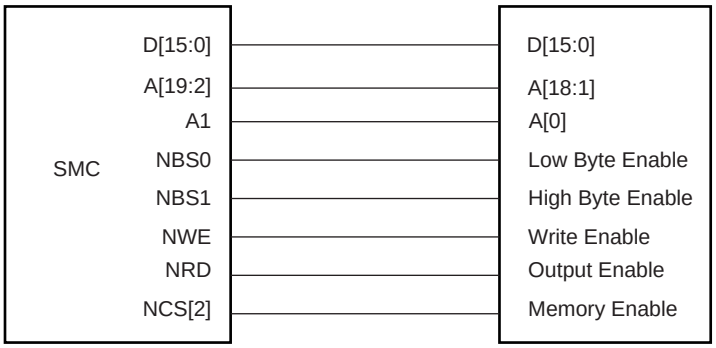
Figure 34-2 shows how to connect a 512-Kbyte × 8-bit memory on NCS2. Figure 34-3 shows how to connect a 512-Kbyte × 16-bit memory on NCS2.

Figure 34-2. Memory Connection for an 8-bit Data Bus



Downloaded from Arrow.com.

Figure 34-3. Memory Connection for a 16-bit Data Bus



34.7.2 Byte Write or Byte Select Access

Each chip select with a 16-bit data bus can operate with one of two different types of write access: byte write or byte select. This is controlled by the BAT field of the SMC_MODE register for the corresponding chip select.

34.7.2.1 Byte Write Access

Byte write access is used to connect 2 × 8-bit devices as a 16-bit memory, and supports one write signal per byte of the data bus and a single read signal.

Note that the SMC does not allow boot in Byte write access mode.

For 16-bit devices, the SMC provides NWR0 and NWR1 write signals for respectively Byte0 (lower byte) and Byte1 (upper byte) of a 16-bit bus. One single read signal (NRD) is provided.

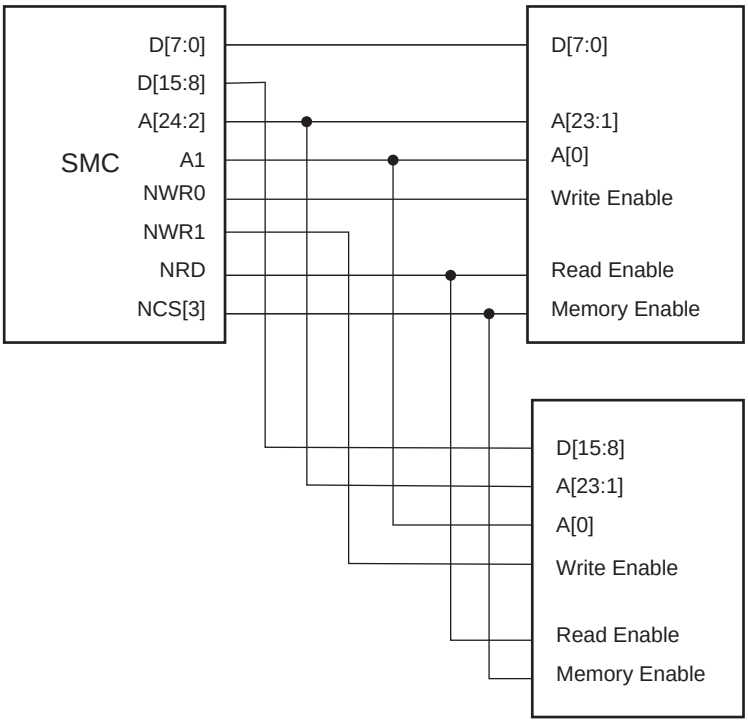
34.7.2.2 Byte Select Access

Byte select access is used to connect one 16-bit device. In this mode, read/write operations can be enabled/disabled at byte level. One byte-select line per byte of the data bus is provided. One NRD and one NWE signal control read and write.

For 16-bit devices, the SMC provides NBS0 and NBS1 selection signals for respectively Byte0 (lower byte) and Byte1 (upper byte) of a 16-bit bus.

Downloaded from Arrow.com.

Figure 34-4. Connection of 2 × 8-bit Devices on a 16-bit Bus: Byte Write Option



34.7.2.3 Signal Multiplexing

Depending on the byte access type (BAT), only the byte write signals or the byte select signals are used. To save IOs at the external bus interface, control signals at the SMC interface are multiplexed. Table 34-4 shows signal multiplexing depending on the data bus width and the byte access type.

For 16-bit devices, bit A0 of address is unused. When the Byte Select option is selected, NWR1 is unused. When the Byte Write option is selected, NBS0 is unused.

Table 34-4. SMC Multiplexed Signal Translation

	Signal Name		
	16-bit Bus		8-bit Bus
Device Type	1 x 16-bit	2 x 8-bit	1 x 8-bit
Byte Access Type (BAT)	Byte Select	Byte Write	–
NBS0_A0	NBS0	–	A0
NWE_NWR0	NWE	NWR0	NWE
NBS1_NWR1	NBS1	NWR1	–
A1	A1	A1	A1

Downloaded from Arrow.com.

34.7.3 NAND Flash Support

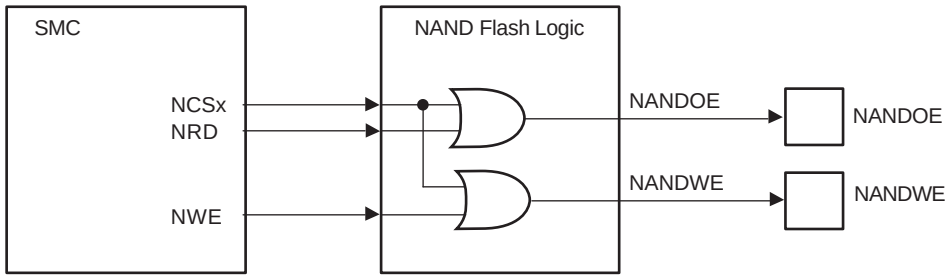
The SMC integrates circuitry that interfaces to NAND Flash devices.

The NAND Flash logic is driven by the SMC. Configuration is done via the SMC_NFCSx field in the CCFG_SMCNFCS register in the Bus Matrix. For details on this register, refer to Section 18. "Bus Matrix (MATRIX)" of this datasheet. The external NAND Flash device is accessed via the address space reserved for the chip select programmed.

The user can connect up to four NAND Flash devices with separate chip selects.

The NAND Flash logic drives the read and write command signals of the SMC on the NANDOE and NANDWE signals when the NCSx programmed is active. NANDOE and NANDWE are disabled as soon as the transfer address fails to lie in the NCSx programmed address space.

Figure 34-5. NAND Flash Signal Multiplexing on SMC Pins



Notes:
1. NCSx is active when CCFG_SMCNFCS.SMC_NFCSx=1.
2. When the NAND Flash logic is activated, (SMC_NFCSx=1), the NWE pin can be used only in Peripheral mode (NWE function). If the NWE function is not used for other external memories (SRAM, LCD), it must be configured in one of the following modes:
   – PIO Input with pull-up enabled (default state after reset)
   – PIO Output set at level 1

The address latch enable and command latch enable signals on the NAND Flash device are driven by address bits A22 and A21of the address bus. Any bit of the address bus can also be used for this purpose. The command, address or data words on the data bus of the NAND Flash device use their own addresses within the NCSx address space (configured in the register CCFG_SMCNFCS in the Bus Matrixe). The chip enable (CE) signal of the device and the ready/busy (R/B) signals are connected to PIO lines. The CE signal then remains asserted even when NAND Flash chip select is not selected, preventing the device from returning to Standby mode. The NANDCS output signal should be used in accordance with the external NAND Flash device type.

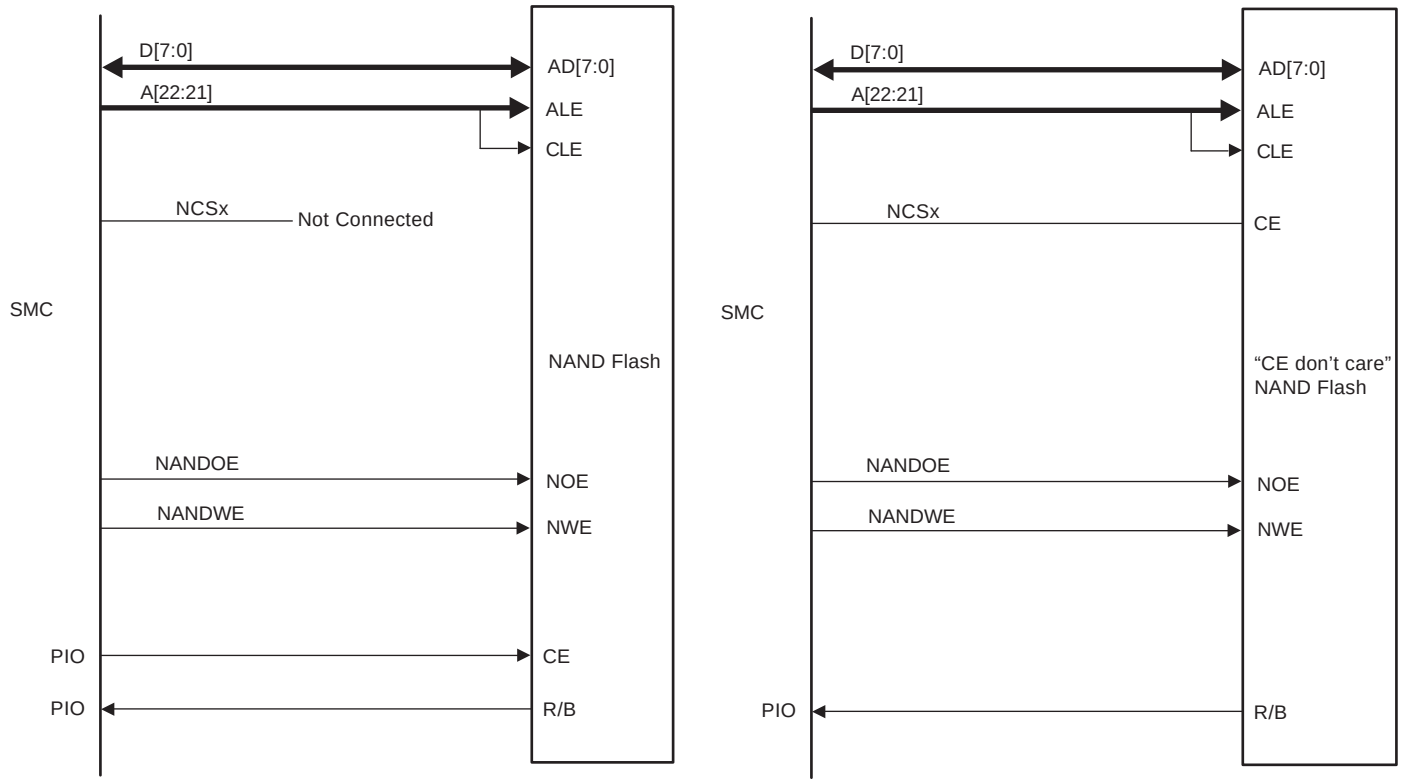
Two types of CE behavior exist depending on the NAND Flash device:

- Standard NAND Flash devices require that the CE pin remains asserted low continuously during the read busy period to prevent the device from returning to Standby mode. Since the SMC asserts the NCSx signal high, it is necessary to connect the CE pin of the NAND Flash device to a GPIO line, in order to hold it low during the busy period preceding data read out.
- This restriction has been removed for “CE don’t care” NAND Flash devices. The NCSx signal can be directly connected to the CE pin of the NAND Flash device.

Figure 34-6 illustrates both topologies: Standard and “CE don’t care” NAND Flash.

Downloaded from Arrow.com.

Figure 34-6. Standard and “CE don’t care” NAND Flash Application Examples



34.8 Application Example

34.8.1 Implementation Examples

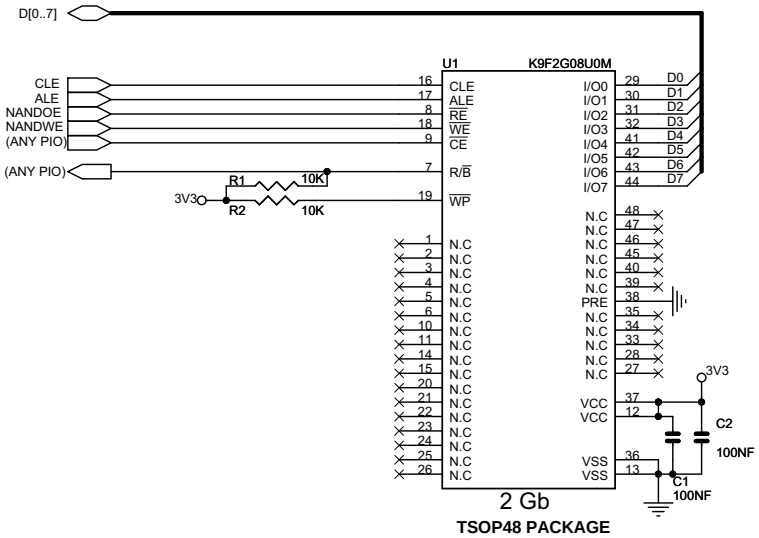
Hardware configurations are given for illustration only. The user should refer to the manufacturer web site to check for memory device availability.

For hardware implementation examples, refer to the evaluation kit schematics for this microcontroller, which show examples of a connection to an LCD module and NAND Flash.

Downloaded from Arrow.com.

34.8.1.1 8-bit NAND Flash

Hardware Configuration



Software Configuration

Perform the following configuration:

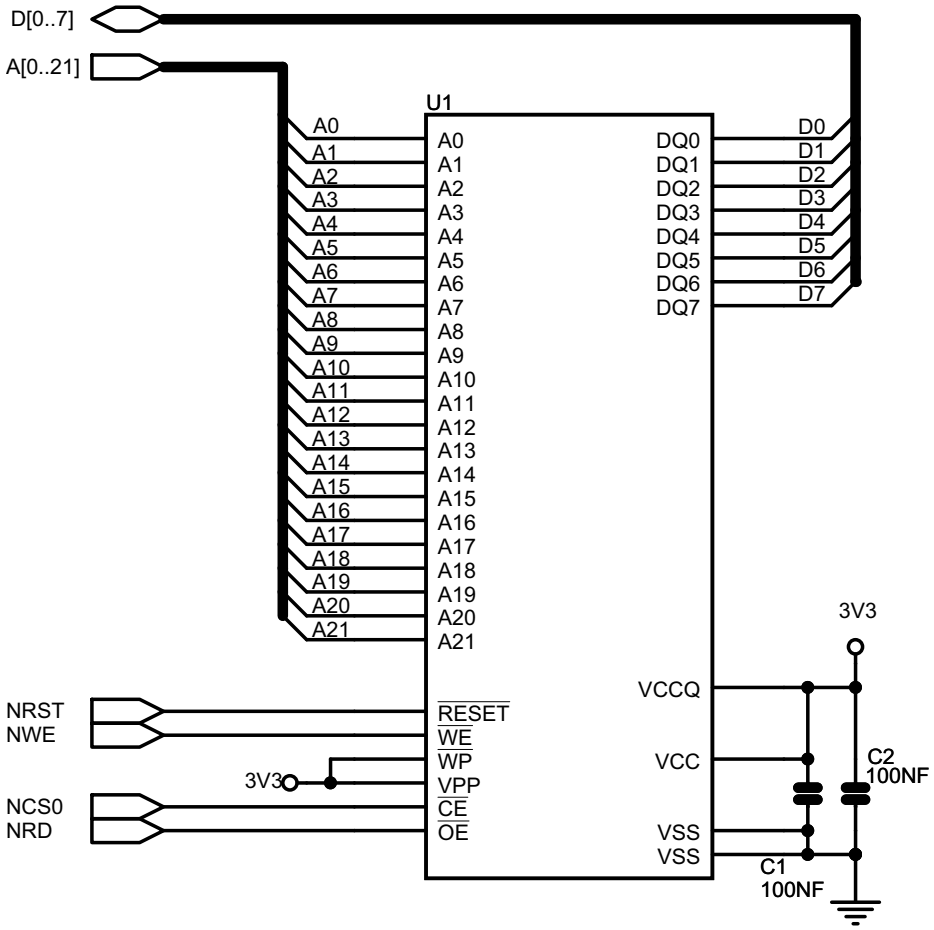
1. Select the chip select used to drive the NAND Flash by setting the bit CCFG_SMCNFCS.SMC_NFCSx.
2. Reserve A21 / A22 for ALE / CLE functions. Address and Command Latches are controlled by setting the address bits A21 and A22, respectively, during accesses.
3. NANDOE and NANDWE signals are multiplexed with PIO lines. Thus, the dedicated PIOs must be programmed in Peripheral mode in the PIO controller.
4. Configure a PIO line as an input to manage the Ready/Busy signal.
5. Configure SMC CS3 Setup, Pulse, Cycle and Mode according to NAND Flash timings, the data bus width and the system bus frequency.

In this example, the NAND Flash is not addressed as a “CE don’t care”. To address it as a “CE don’t care”, connect NCS3 (if SMC_NFCS3 is set) to the NAND Flash CE.

Downloaded from Arrow.com.

34.8.1.2 NOR Flash

Hardware Configuration



Software Configuration

Configure the SMC CS0 Setup, Pulse, Cycle and Mode depending on Flash timings and system bus frequency.

34.9 Standard Read and Write Protocols

In the following sections, the byte access type is not considered. Byte select lines (NBS0 to NBS1) always have the same timing as the A address bus. NWE represents either the NWE signal in byte select access type or one of the byte write lines (NWR0 to NWR1) in byte write access type. NWR0 to NWR1 have the same timings and protocol as NWE. If D[15:8] are used, they have the same timing as D[7:0]. In the same way, NCS represents one of the NCS[0..3] chip select lines.

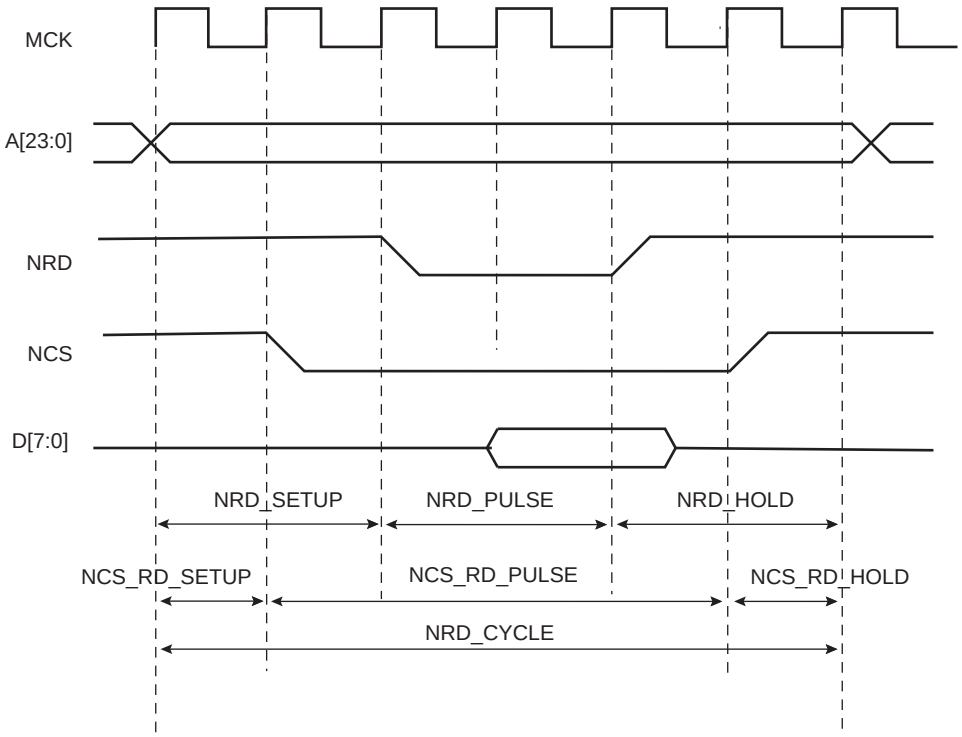
Downloaded from Arrow.com.

34.9.1 Read Waveforms

The read cycle is shown in Figure 34-7.

The read cycle starts with the address setting on the memory address bus.

Figure 34-7. Standard Read Cycle



34.9.1.1 NRD Waveform

The NRD signal is characterized by a setup timing, a pulse width and a hold timing.

- NRD_SETUP— NRD setup time is defined as the setup of address before the NRD falling edge;
- NRD_PULSE—NRD pulse length is the time between NRD falling edge and NRD rising edge;
- NRD_HOLD—NRD hold time is defined as the hold time of address after the NRD rising edge.

34.9.1.2 NCS Waveform

The NCS signal can be divided into a setup time, pulse length and hold time:

- NCS_RD_SETUP—NCS setup time is defined as the setup time of address before the NCS falling edge.
- NCS_RD_PULSE—NCS pulse length is the time between NCS falling edge and NCS rising edge;
- NCS_RD_HOLD—NCS hold time is defined as the hold time of address after the NCS rising edge.

34.9.1.3 Read Cycle

The NRD_CYCLE time is defined as the total duration of the read cycle, i.e., from the time where address is set on the address bus to the point where address may change. The total read cycle time is defined as:

NRD_CYCLE = NRD_SETUP + NRD_PULSE + NRD_HOLD,

as well as

NRD_CYCLE = NCS_RD_SETUP + NCS_RD_PULSE + NCS_RD_HOLD

All NRD and NCS timings are defined separately for each chip select as an integer number of Master Clock cycles. The NRD_CYCLE field is common to both the NRD and NCS signals, thus the timing period is of the same duration.

Downloaded from Arrow.com.

NRD_CYCLE, NRD_SETUP, and NRD_PULSE implicitly define the NRD_HOLD value as:

NRD_HOLD = NRD_CYCLE - NRD SETUP - NRD PULSE

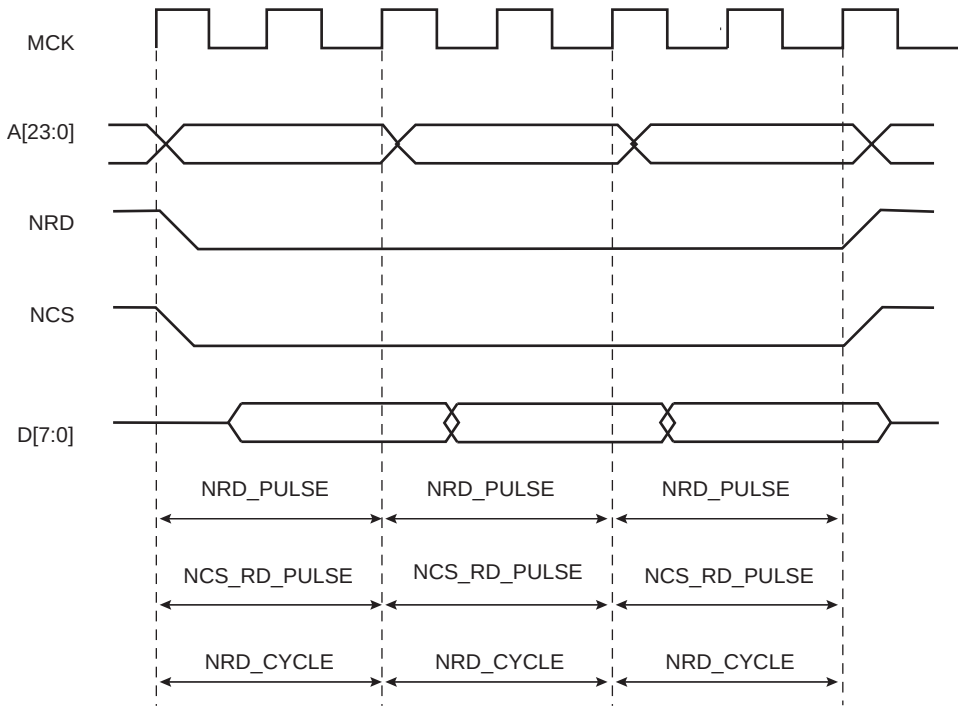
NRD_CYCLE, NCS_RD_SETUP, and NCS_RD_PULSE implicitly define the NCS_RD_HOLD value as:

NCS_RD_HOLD = NRD_CYCLE - NCS_RD_SETUP - NCS_RD_PULSE

34.9.1.4 Null Delay Setup and Hold

If null setup and hold parameters are programmed for NRD and/or NCS, NRD and NCS remain active continuously in case of consecutive read cycles in the same memory (see Figure 34-8).

Figure 34-8. No Setup, No Hold on NRD and NCS Read Signals



34.9.1.5 Null Pulse

Programming a null pulse is not permitted. The pulse must be at least set to 1. A null value leads to unpredictable behavior.

34.9.2 Read Mode

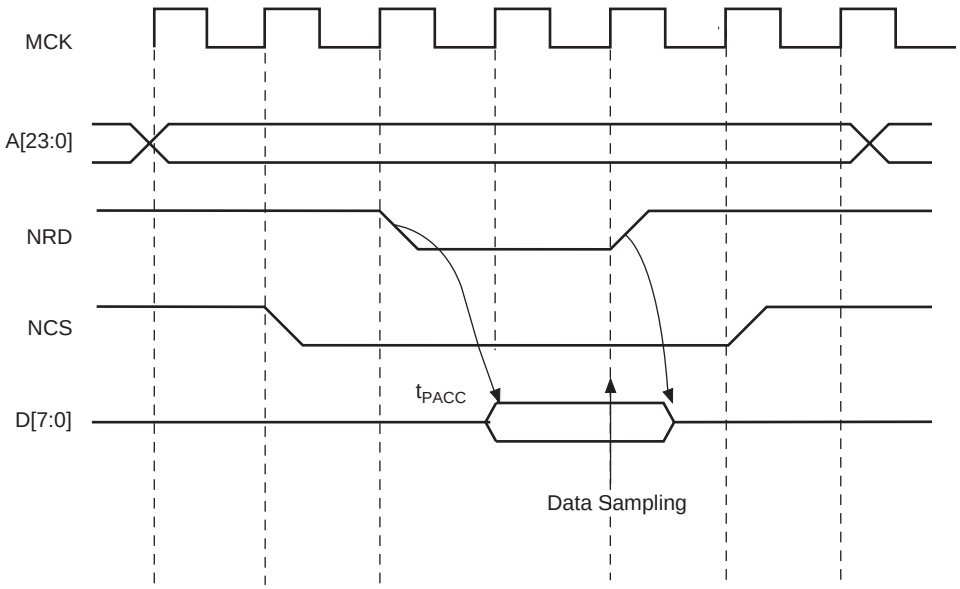
As NCS and NRD waveforms are defined independently of one other, the SMC needs to know when the read data is available on the data bus. The SMC does not compare NCS and NRD timings to know which signal rises first. The READ_MODE bit in the SMC_MODE register of the corresponding chip select indicates which signal of NRD and NCS controls the read operation.

Downloaded from Arrow.com.

34.9.2.1 Read is Controlled by NRD (SMC_MODE.READ_MODE = 1):

Figure 34-9 shows the waveforms of a read operation of a typical asynchronous RAM. The read data is available t_{PACC} after the falling edge of NRD, and turns to 'Z' after the rising edge of NRD. In this case, SMC_MODE.READ_MODE must be set to 1 (read is controlled by NRD), to indicate that data is available with the rising edge of NRD. The SMC samples the read data internally on the rising edge of Master Clock that generates the rising edge of NRD, whatever the programmed waveform of NCS may be.

Figure 34-9. SMC_MODE.READ_MODE = 1: Data is sampled by SMC before the rising edge of NRD

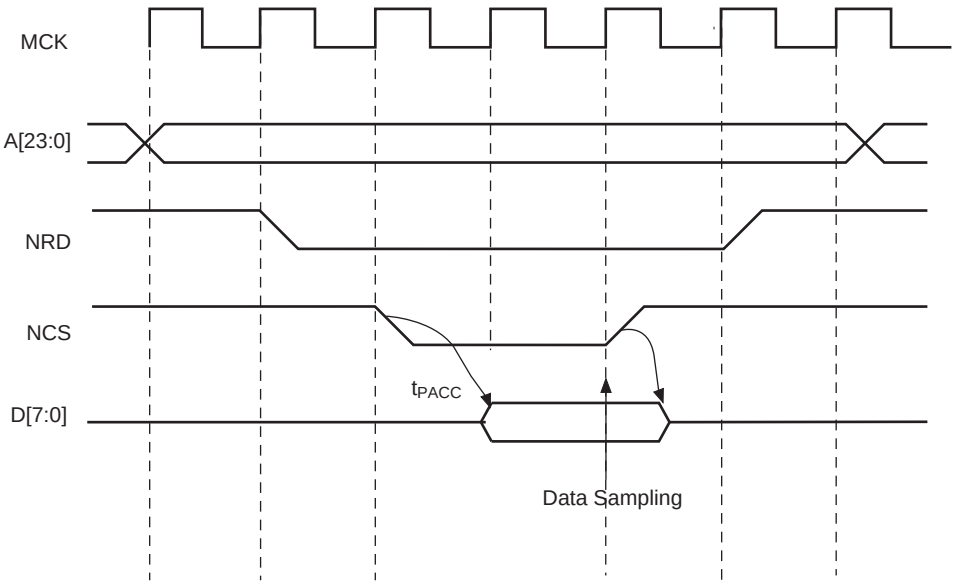


Downloaded from Arrow.com.

34.9.2.2 Read is Controlled by NCS (SMC_MODE.READ_MODE = 0)

Figure 34-10 shows the typical read cycle of an LCD module. The read data is valid t_{PACC} after the falling edge of the NCS signal and remains valid until the rising edge of NCS. Data must be sampled when NCS is raised. In this case, the SMC_MODE.READ_MODE must be set to 0 (read is controlled by NCS): the SMC internally samples the data on the rising edge of Master Clock that generates the rising edge of NCS, whatever the programmed waveform of NRD may be.

Figure 34-10. SMC_MODE.READ_MODE = 0: Data is Sampled by SMC Before the Rising Edge of NCS



Downloaded from Arrow.com.

34.9.3 Write Waveforms

The write protocol is similar to the read protocol. It is depicted in Figure 34-11. The write cycle starts with the address setting on the memory address bus.

34.9.3.1 NWE Waveforms

The NWE signal is characterized by a setup timing, a pulse width and a hold timing.

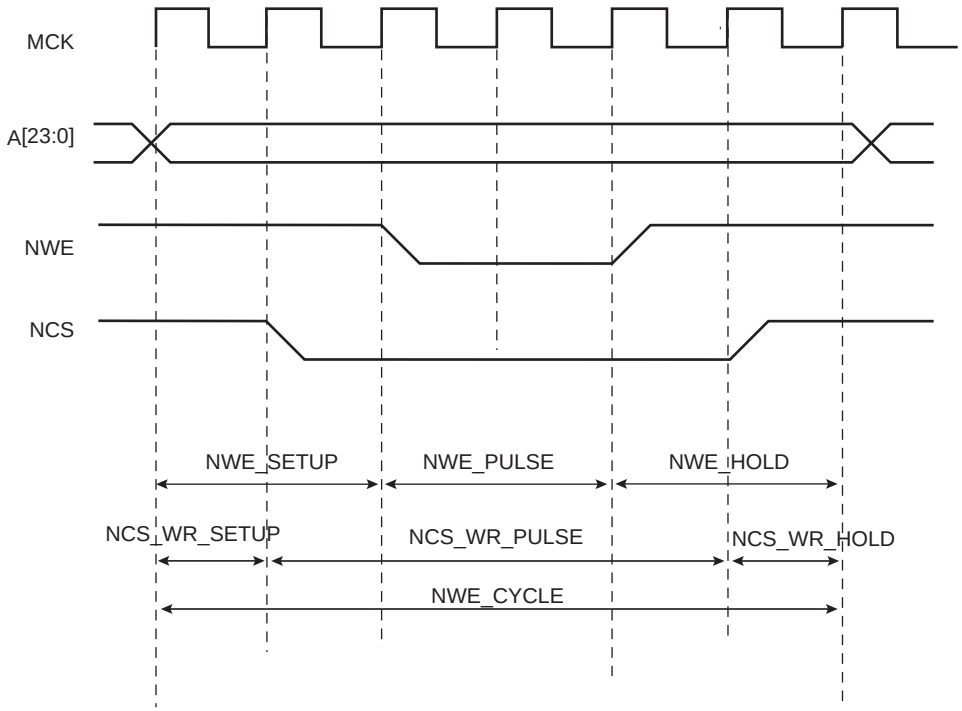
- NWE_SETUP—the NWE setup time is defined as the setup of address and data before the NWE falling edge;
- NWE_PULSE—the NWE pulse length is the time between NWE falling edge and NWE rising edge;
- NWE_HOLD—the NWE hold time is defined as the hold time of address and data after the NWE rising edge.

34.9.3.2 NCS Waveforms

The NCS signal waveforms in write operation are not the same that those applied in read operations, but are separately defined:

- NCS_WR_SETUP—the NCS setup time is defined as the setup time of address before the NCS falling edge.
- NCS_WR_PULSE—the NCS pulse length is the time between NCS falling edge and NCS rising edge;
- NCS_WR_HOLD—the NCS hold time is defined as the hold time of address after the NCS rising edge.

Figure 34-11. Write Cycle



34.9.3.3 Write Cycle

The write_cycle time is defined as the total duration of the write cycle; that is, from the time where address is set on the address bus to the point where address may change. The total write cycle time is defined as:

NWE_CYCLE = NWE_SETUP + NWE_PULSE + NWE_HOLD,

as well as

NWE_CYCLE = NCS_WR_SETUP + NCS_WR_PULSE + NCS_WR_HOLD

Downloaded from Arrow.com.

All NWE and NCS (write) timings are defined separately for each chip select as an integer number of Master Clock cycles. The NWE_CYCLE field is common to both the NWE and NCS signals, thus the timing period is of the same duration.

NWE_CYCLE, NWE_SETUP, and NWE_PULSE implicitly define the NWE_HOLD value as:

NWE_HOLD = NWE_CYCLE - NWE_SETUP - NWE_PULSE

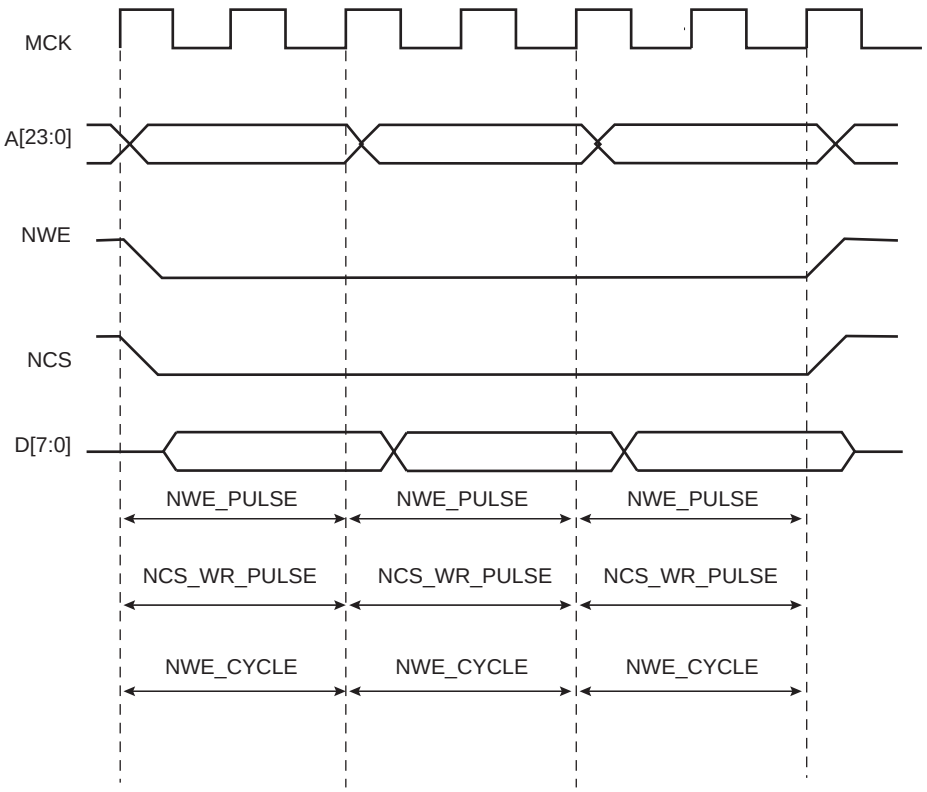
NWE_CYCLE, NCS_WR_SETUP, and NCS_WR_PULSE implicitly define the NCS_WR_HOLD value as:

NCS_WR_HOLD = NWE_CYCLE - NCS_WR_SETUP - NCS_WR_PULSE

34.9.3.4 Null Delay Setup and Hold

If null setup parameters are programmed for NWE and/or NCS, NWE and/or NCS remain active continuously in case of consecutive write cycles in the same memory (see Figure 34-12). However, for devices that perform write operations on the rising edge of NWE or NCS, such as SRAM, either a setup or a hold must be programmed.

Figure 34-12. Null Setup and Hold Values of NCS and NWE in Write Cycle



34.9.3.5 Null Pulse

Programming null pulse is not permitted. Pulse must be at least set to 1. A null value leads to unpredictable behavior.

Downloaded from Arrow.com.

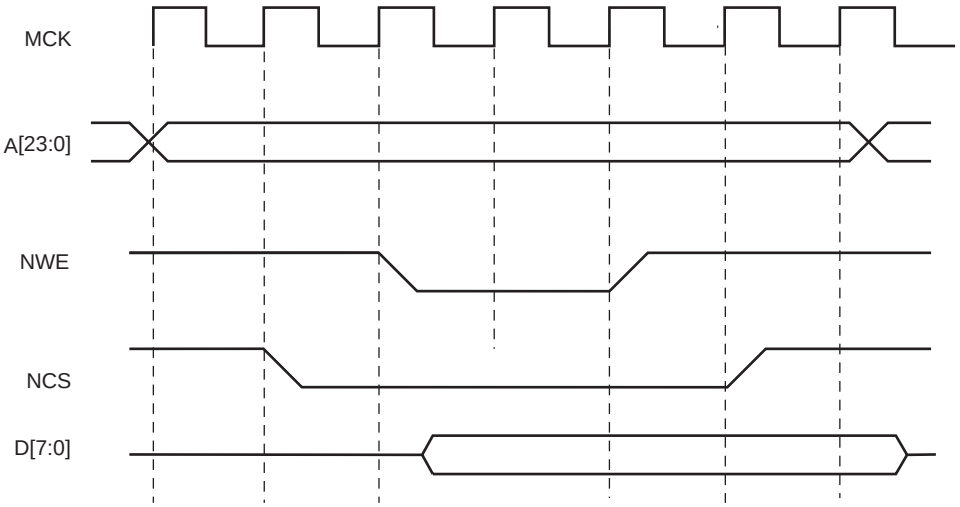
34.9.4 Write Mode

The bit WRITE_MODE in the SMC_MODE register of the corresponding chip select indicates which signal controls the write operation.

34.9.4.1 Write is Controlled by NWE (SMC.MODE.WRITE_MODE = 1):

Figure 34-13 shows the waveforms of a write operation with SMC_MODE.WRITE_MODE set . The data is put on the bus during the pulse and hold steps of the NWE signal. The internal data buffers are switched to Output mode after the NWE_SETUP time, and until the end of the write cycle, regardless of the programmed waveform on NCS.

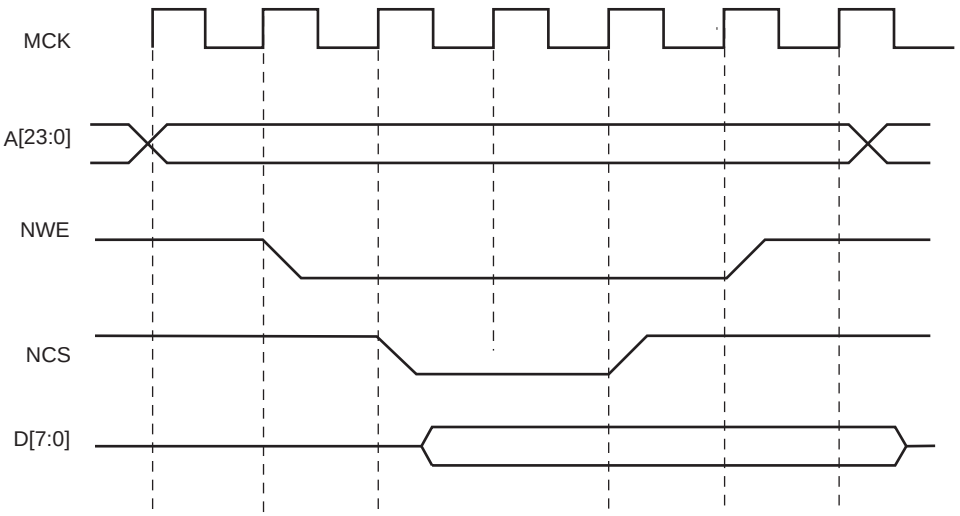
Figure 34-13. SMC_MODE.WRITE_MODE = 1. Write Operation is Controlled by NWE



34.9.4.2 Write is Controlled by NCS (SMC.MODE.WRITE_MODE = 0)

Figure 34-14 shows the waveforms of a write operation with SMC_MODE.WRITE_MODE cleared. The data is put on the bus during the pulse and hold steps of the NCS signal. The internal data buffers are switched to Output mode after the NCS_WR_SETUP time, and until the end of the write cycle, regardless of the programmed waveform on NWE.

Figure 34-14. WRITE_MODE = 0. Write Operation is Controlled by NCS



Downloaded from Arrow.com.

34.9.5 Register Write Protection

To prevent any single software error that may corrupt SMC behavior, the registers listed below can be write-protected by setting the WPEN bit in the SMC Write Protection Mode register (SMC_WPMR).

If a write access in a write-protected register is detected, the WPVS flag in the SMC Write Protection Status register (SMC_WPSR) is set and the field WPVSRC indicates in which register the write access has been attempted.

The WPVS flag is automatically cleared after reading the SSMC_WPSR.

The following registers can be write-protected:

- "SMC Setup Register"
- "SMC Pulse Register"
- "SMC Cycle Register"
- "SMC Mode Register"

34.9.6 Coding Timing Parameters

All timing parameters are defined for one chip select and are grouped together in one register according to their type.

The SMC_SETUP register groups the definition of all setup parameters:

- NRD_SETUP
- NCS_RD_SETUP
- NWE_SETUP
- NCS_WR_SETUP

The SMC_PULSE register groups the definition of all pulse parameters:

- NRD_PULSE
- NCS_RD_PULSE
- NWE_PULSE
- NCS_WR_PULSE

The SMC_CYCLE register groups the definition of all cycle parameters:

- NRD_CYCLE
- NWE_CYCLE

Table 34-5 shows how the timing parameters are coded and their permitted range.

Table 34-5. Coding and Range of Timing Parameters

Coded Value	Number of Bits	Effective Value	Permitted Range	
			Coded Value	Effective Value
setup [5:0]	6	128 × setup[5] + setup[4:0]	0 ≤ 31	0 ≤ 128+31
pulse [6:0]	7	256 × pulse[6] + pulse[5:0]	0 ≤ 63	0 ≤ 256+63
cycle [8:0]	9	256 × cycle[8:7] + cycle[6:0]	0 ≤ 127	0 ≤ 256+127 0 ≤ 512+127 0 ≤ 768+127

Downloaded from Arrow.com.

34.9.7 Reset Values of Timing Parameters

Table 34-6 gives the default value of timing parameters at reset.

Table 34-6. Reset Values of Timing Parameters

Parameter	Reset Value	Definition
SMC_SETUP	0x01010101	All setup timings are set to 1.
SMC_PULSE	0x01010101	All pulse timings are set to 1.
SMC_CYCLE	0x00030003	The read and write operations continue for 3 Master Clock cycles and provide one hold cycle.
WRITE_MODE	1	Write is controlled with NWE.
READ_MODE	1	Read is controlled with NRD.

34.9.8 Usage Restriction

The SMC does not check the validity of the user-programmed parameters. If the sum of SETUP and PULSE parameters is larger than the corresponding CYCLE parameter, this leads to unpredictable behavior of the SMC.

- For read operations:

  Null but positive setup and hold of address and NRD and/or NCS can not be guaranteed at the memory interface because of the propagation delay of theses signals through external logic and pads. If positive setup and hold values must be verified, then it is strictly recommended to program non-null values so as to cover possible skews between address, NCS and NRD signals.

- For write operations:

  If a null hold value is programmed on NWE, the SMC can guarantee a positive hold of address and NCS signal after the rising edge of NWE. This is true for SMC_MODE.WRITE_MODE = 1 only. See Section 34.11.2 ”Early Read Wait State”.

- For read and write operations:

  A null value for pulse parameters is forbidden and may lead to unpredictable behavior.

  In read and write cycles, the setup and hold time parameters are defined in reference to the address bus. For external devices that require setup and hold time between NCS and NRD signals (read), or between NCS and NWE signals (write), these setup and hold times must be converted into setup and hold times in reference to the address bus.

Downloaded from Arrow.com.

34.10 Scrambling/Unscrambling Function

The external data bus can be scrambled to prevent recovery of intellectual property data located in off-chip memories by means of data analysis at the package pin level of either the microcontroller or the memory device.

The scrambling and unscrambling are performed on-the-fly without additional wait states.

The scrambling/unscrambling function can be enabled or disabled by configuring the CSxSE bits in the SMC Off-Chip Memory Scrambling Register (SMC_OCMS).

When multiple chip selects are handled, the scrambling function per chip select is configurable using the CSxSE bits in the SMC_OCMS register.

The scrambling method depends on two user-configurable key registers, SMC_KEY1 and SMC_KEY2 plus a random value depending on device processing characteristics. These key registers cannot be read. They can be written once after a system reset.

The scrambling user key or the seed for key generation must be securely stored in a reliable non-volatile memory in order to recover data from the off-chip memory. Any data scrambled with a given key cannot be recovered if the key is lost.

34.11 Automatic Wait States

Under certain circumstances, the SMC automatically inserts idle cycles between accesses to avoid bus contention or operation conflict.

34.11.1 Chip Select Wait States

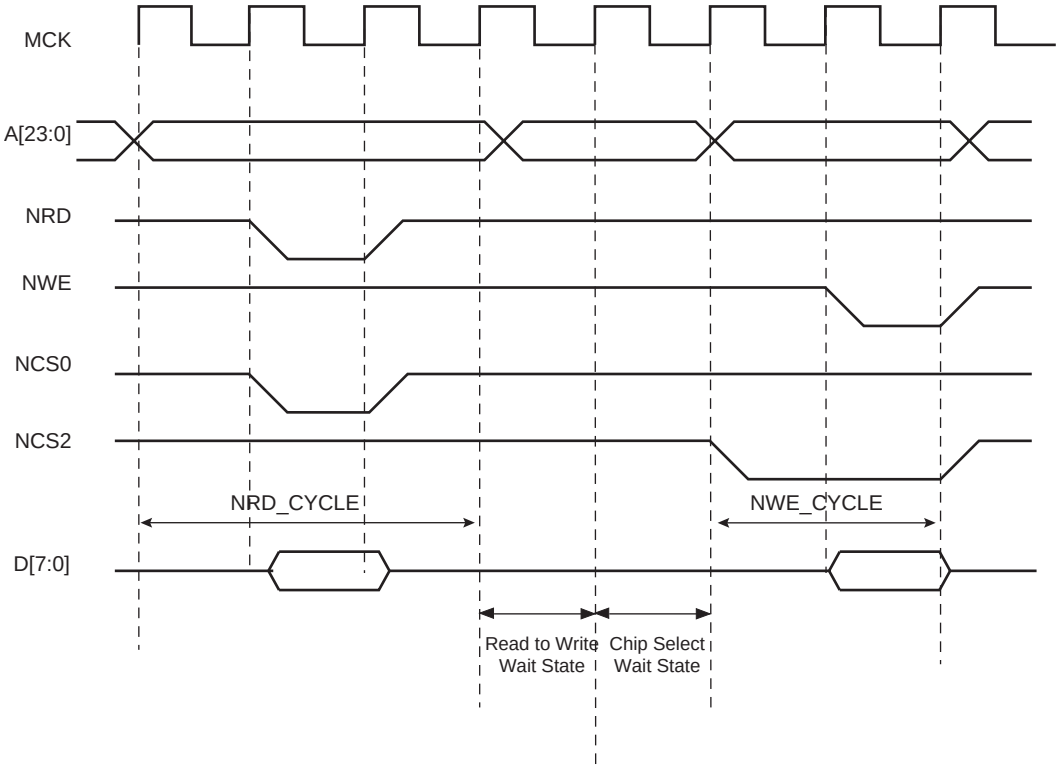
The SMC always inserts an idle cycle between two transfers on separate chip selects. This idle cycle ensures that there is no bus contention between the de-activation of one device and the activation of the next one.

During chip select wait state, all control lines are turned inactive: NWR, NCS[0..3], NRD lines are all set to 1.

Figure 34-15 illustrates a chip select wait state between access on chip select 0 and chip select 2.

Downloaded from Arrow.com.

Figure 34-15. Chip Select Wait State between a Read Access on NCS0 and a Write Access on NCS2



34.11.2 Early Read Wait State

In some cases, the SMC inserts a wait state cycle between a write access and a read access to allow time for the write cycle to end before the subsequent read cycle begins. This wait state is not generated in addition to a chip select wait state. The early read cycle thus only occurs between a write and read access to the same memory device (same chip select).

An early read wait state is automatically inserted if at least one of the following conditions is valid:

- if the write controlling signal has no hold time and the read controlling signal has no setup time (Figure 34-16).
- in NCS Write controlled mode (SMC_MODE.WRITE_MODE = 0), if there is no hold timing on the NCS signal and the NCS_RD_SETUP parameter is set to 0, regardless of the Read mode (Figure 34-17). The write operation must end with a NCS rising edge. Without an Early Read Wait State, the write operation could not complete properly.
- in NWE controlled mode (SMC_MODE.WRITE_MODE = 1) and if there is no hold timing (NWE_HOLD = 0), the feedback of the write control signal is used to control address, data, and chip select lines. If the external write control signal is not inactivated as expected due to load capacitances, an Early Read Wait State is inserted and address, data and control signals are maintained one more cycle. See Figure 34-18.

Downloaded from Arrow.com.

Figure 34-16. Early Read Wait State: Write with No Hold Followed by Read with No Setup

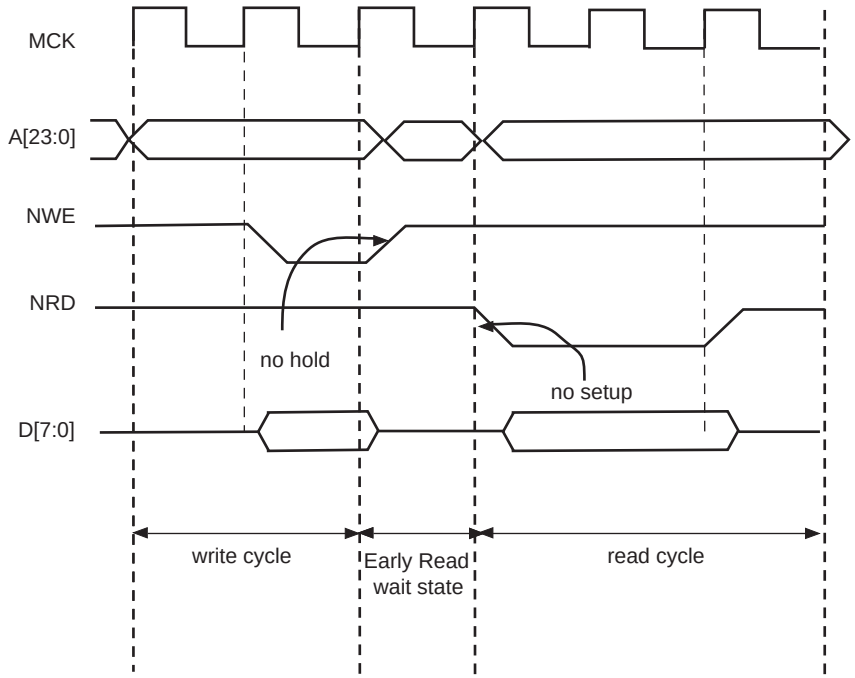
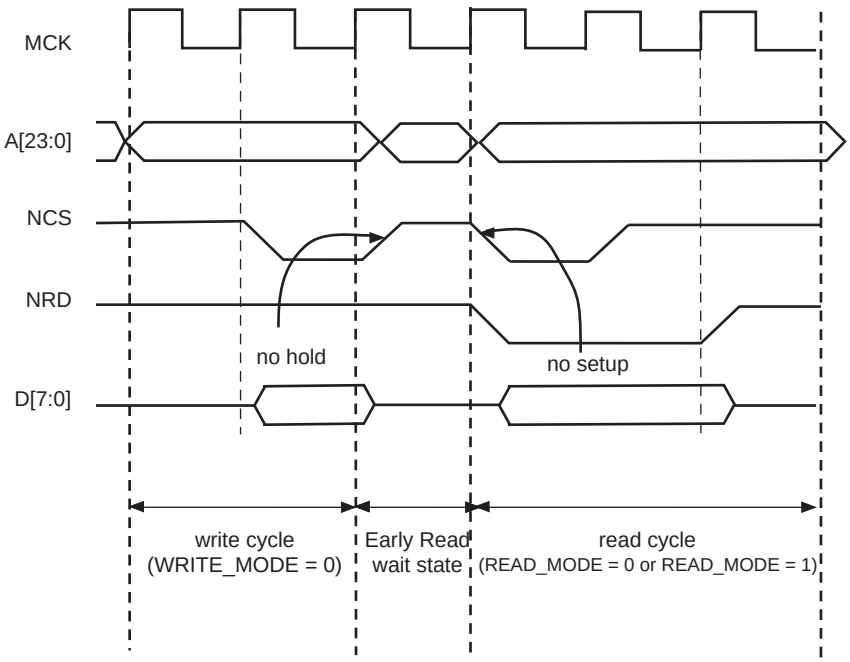
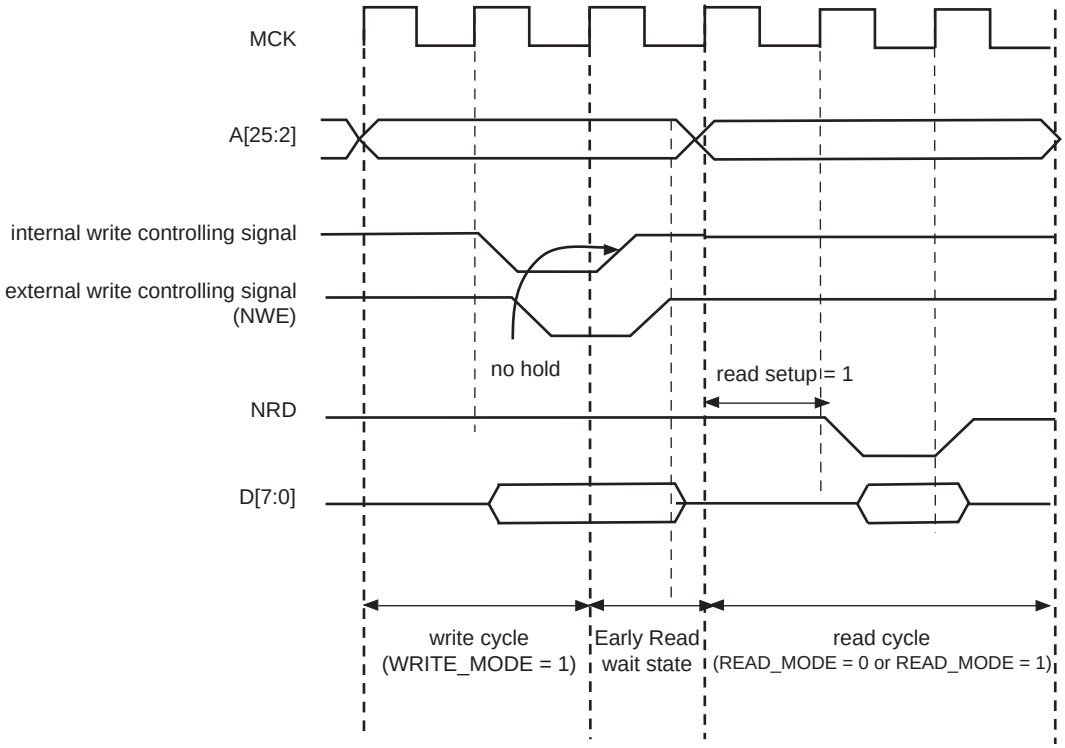


Figure 34-17. Early Read Wait State: NCS-controlled write with no hold followed by a read with no NCS setup



Downloaded from Arrow.com.

Figure 34-18. Early Read Wait State: NWE-controlled write with no hold followed by a read with one set-up cycle



34.11.3 Reload User Configuration Wait State

The user may change any of the configuration parameters by writing the SMC user interface.

When detecting that a new user configuration has been written in the user interface, the SMC inserts a wait state before starting the next access. This “reload user configuration wait state” is used by the SMC to load the new set of parameters to apply to next accesses.

The reload configuration wait state is not applied in addition to the chip select wait state. If accesses before and after re-programming the user interface are made to different devices (chip selects), then one single chip select wait state is applied.

On the other hand, if accesses before and after writing the user interface are made to the same device, a reload configuration wait state is inserted, even if the change does not concern the current chip select.

34.11.3.1 User Procedure

To insert a reload configuration wait state, the SMC detects a write access to any SMC_MODE register of the user interface. If the user only modifies timing registers (SMC_SETUP, SMC_PULSE, SMC_CYCLE registers) in the user interface, he must validate the modification by writing the SMC_MODE, even if no change was made on the mode parameters.

The user must not change the configuration parameters of an SMC chip select (Setup, Pulse, Cycle, Mode) if accesses are performed on this CS during the modification. Any change of the chip select parameters, while fetching the code from a memory connected on this CS, may lead to unpredictable behavior. The instructions used to modify the parameters of an SMC chip select can be executed from the internal RAM or from a memory connected to another CS.

Downloaded from Arrow.com.

34.11.3.2 Slow Clock Mode Transition

A reload configuration wait state is also inserted when the Slow Clock mode is entered or exited, after the end of the current transfer (see Section 34.14 ”Slow Clock Mode”).

34.11.4 Read to Write Wait State

Due to an internal mechanism, a wait cycle is always inserted between consecutive read and write SMC accesses.

This wait cycle is referred to as a read to write wait state in this document.

This wait cycle is applied in addition to chip select and reload user configuration wait states when they are to be inserted. See Figure 34-15.

Downloaded from Arrow.com.

34.12 Data Float Wait States

Some memory devices are slow to release the external bus. For such devices, it is necessary to add wait states (data float wait states) after a read access:

- before starting a read access to a different external memory
- before starting a write access to the same device or to a different external one.

The data float output time (t_{DF}) for each external memory device is programmed in the SMC_MODE.TDF_CYCLES field for the corresponding chip select. The value of SMC_MODE.TDF_CYCLES indicates the number of data float wait cycles (between 0 and 15) before the external device releases the bus, and represents the time allowed for the data output to go to high impedance after the memory is disabled.

Data float wait states do not delay internal memory accesses. Hence, a single access to an external memory with long t_{DF} will not slow down the execution of a program from internal memory.

The data float wait states management depends on SMC_MODE.READ_MODE and the SMC_MODE.TDF_MODE fields for the corresponding chip select.

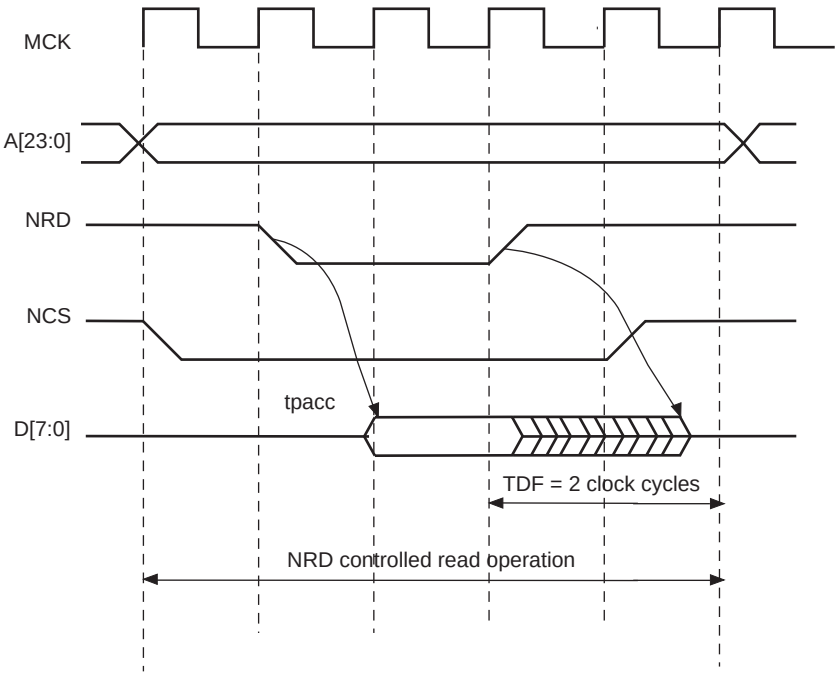
34.12.1 SMC_MODE.READ_MODE

Setting SMC_MODE.READ_MODE to 1 indicates to the SMC that the NRD signal is responsible for turning off the tri-state buffers of the external memory device. The Data Float Period then begins after the rising edge of the NRD signal and lasts SMC_MODE.TDF_CYCLES MCK cycles.

When the read operation is controlled by the NCS signal (SMC_MODE.READ_MODE = 0), the TDF field gives the number of MCK cycles during which the data bus remains busy after the rising edge of NCS.

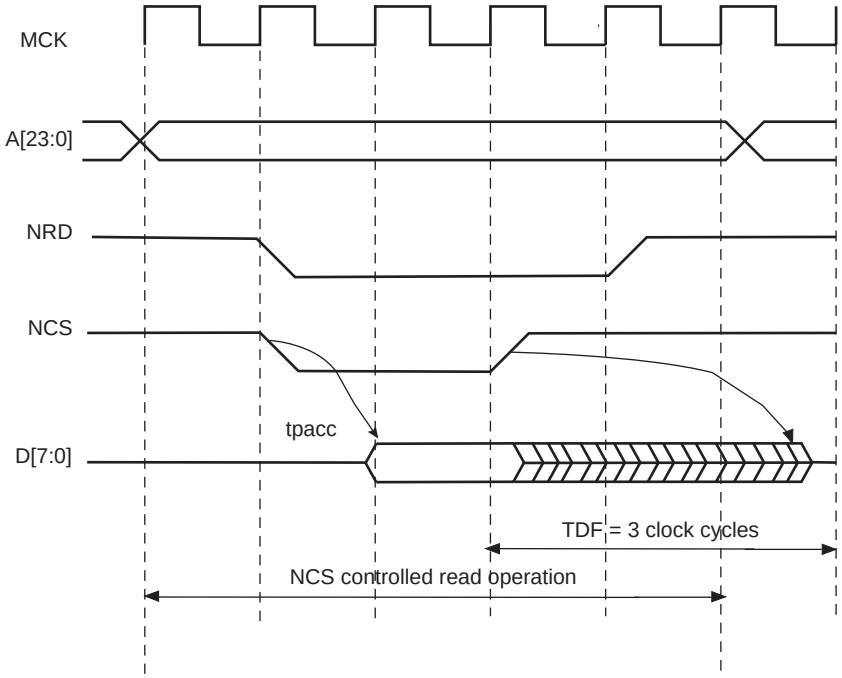
Figure 34-19 illustrates the Data Float Period in NRD-controlled mode (SMC_MODE.READ_MODE =1), assuming a data float period of 2 cycles (SMC_MODE.TDF_CYCLES = 2). Figure 34-20 shows the read operation when controlled by NCS (SMC_MODE.READ_MODE = 0) and SMC_MODE.TDF_CYCLES = 3.

Figure 34-19. TDF Period in NRD Controlled Read Access (TDF = 2)



Downloaded from Arrow.com.

Figure 34-20. TDF Period in NCS Controlled Read Operation (TDF = 3)



Downloaded from Arrow.com.

34.12.2 TDF Optimization Enabled (SMC_MODE.TDF_MODE = 1)

When SMC_MODE.TDF_MODE is set to 1 (TDF optimization is enabled), the SMC takes advantage of the setup period of the next access to optimize the number of wait states cycle to insert.

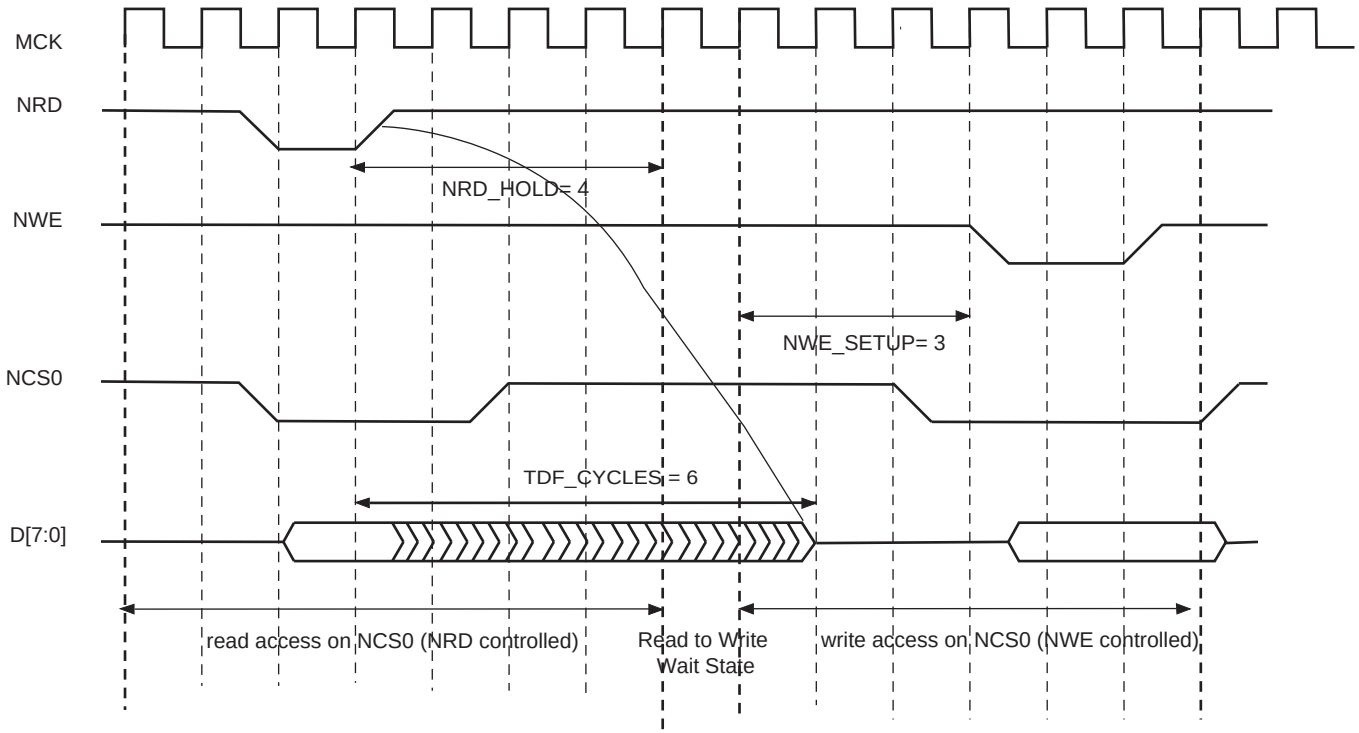
Figure 34-21 shows a read access controlled by NRD, followed by a write access controlled by NWE, on chip select 0. Chip select 0 has been programmed with:

NRD_HOLD = 4; SMC_MODE.READ_MODE = 1 (NRD controlled)

NWE_SETUP = 3; SMC_MODE.WRITE_MODE = 1 (NWE controlled)

SMC_MODE.TDF_CYCLES = 6; SMC_MODE.TDF_MODE = 1 (optimization enabled).

Figure 34-21. TDF Optimization: No TDF wait states are inserted if the TDF period is over when the next access begins



Downloaded from Arrow.com.

34.12.3 TDF Optimization Disabled (SMC_MODE.TDF_MODE = 0)

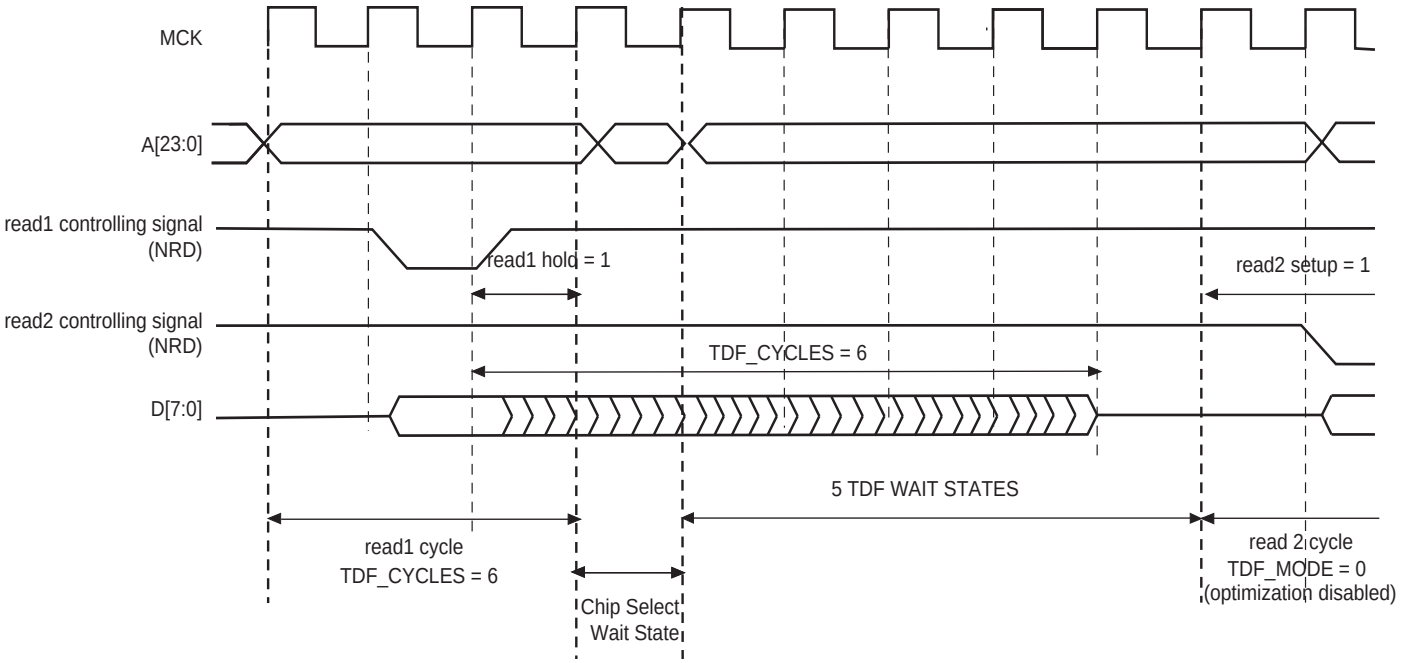
When optimization is disabled, TDF wait states are inserted at the end of the read transfer, so that the data float period is ended when the second access begins. If the hold period of the read1 controlling signal overlaps the data float period, no additional tdf wait states will be inserted.

Figure 34-22, Figure 34-23 and Figure 34-24 illustrate the cases:

- read access followed by a read access on another chip select,
- read access followed by a write access on another chip select,
- read access followed by a write access on the same chip select,

with no TDF optimization.

Figure 34-22. TDF Optimization Disabled (TDF Mode = 0): TDF wait states between 2 read accesses on different chip selects



Downloaded from Arrow.com.

Figure 34-23. TDF Mode = 0: TDF wait states between a read and a write access on different chip selects

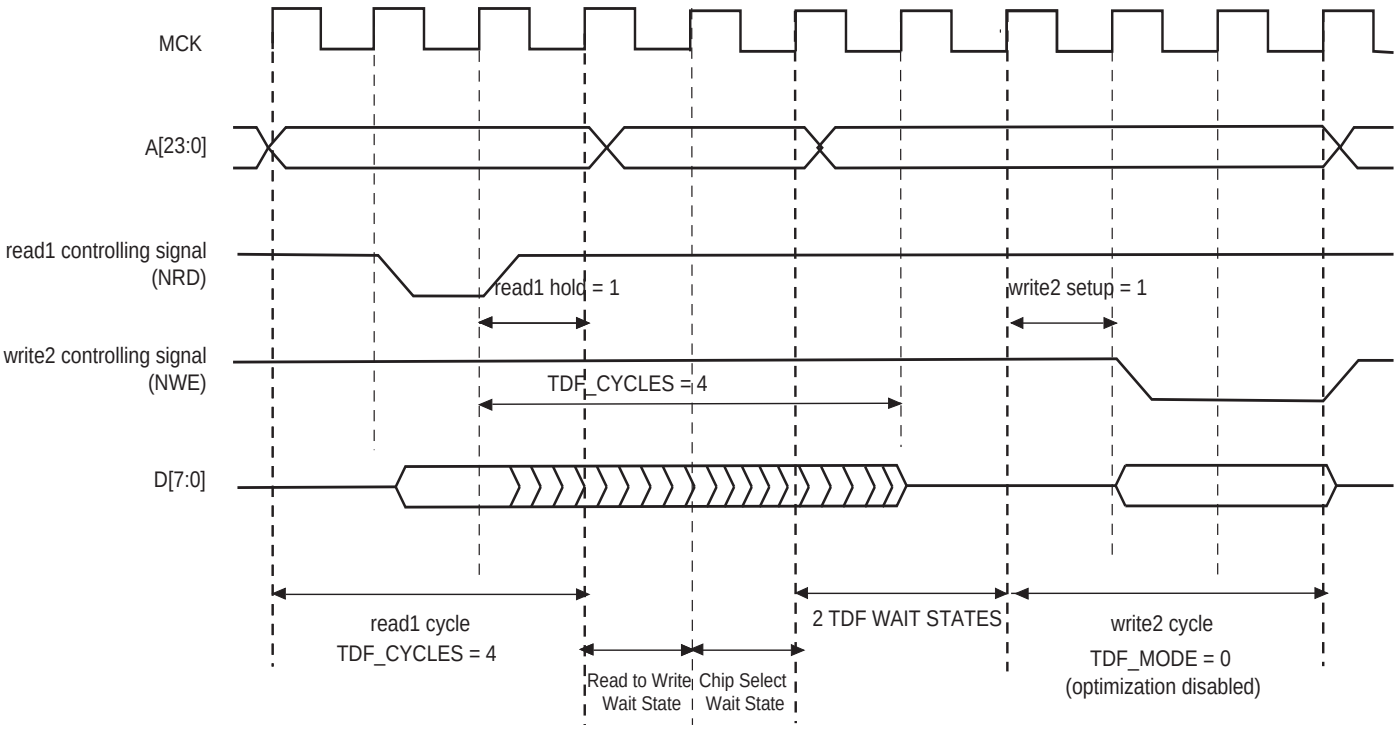
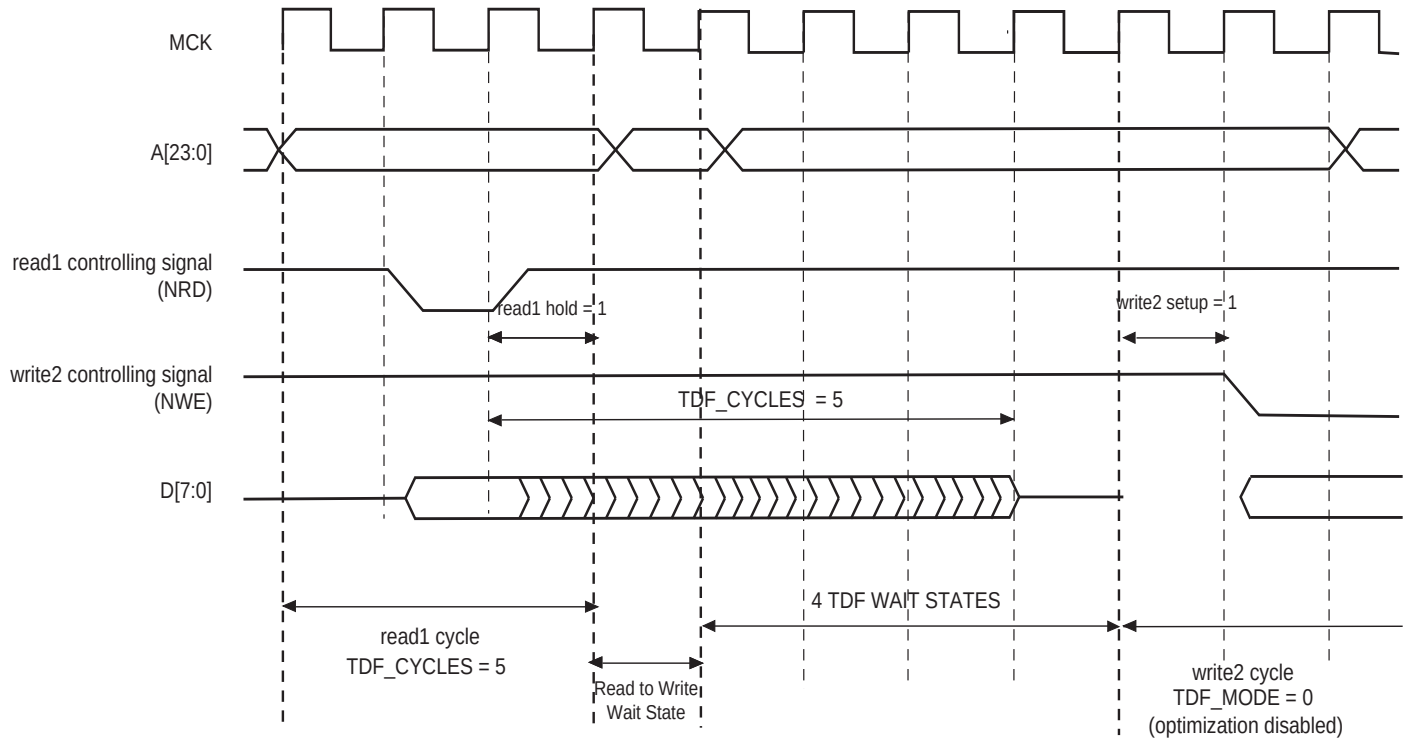


Figure 34-24. TDF Mode = 0: TDF wait states between read and write accesses on the same chip select



Downloaded from Arrow.com.

34.13 External Wait

Any access can be extended by an external device using the NWAIT input signal of the SMC. The SMC_MODE.EXNW_MODE field on the corresponding chip select must be set either to “10” (Frozen mode) or “11” (Ready mode). When SMC_MODE.EXNW_MODE is set to “00” (disabled), the NWAIT signal is simply ignored on the corresponding chip select. The NWAIT signal delays the read or write operation in regards to the read or write controlling signal, depending on the Read and Write modes of the corresponding chip select.

34.13.1 Restriction

When SMC_MODE.EXNW_MODE is enabled, **it is mandatory to program at least one hold cycle for the read/write controlling signal. For that reason, the NWAIT signal cannot be used in Page mode (Section 34.15 ”Asynchronous Page Mode”), or in Slow clock mode (Section 34.14 ”Slow Clock Mode”).**

The NWAIT signal is assumed to be a response of the external device to the read/write request of the SMC. Then NWAIT is examined by the SMC only in the pulse state of the read or write controlling signal. The assertion of the NWAIT signal outside the expected period has no impact on SMC behavior.

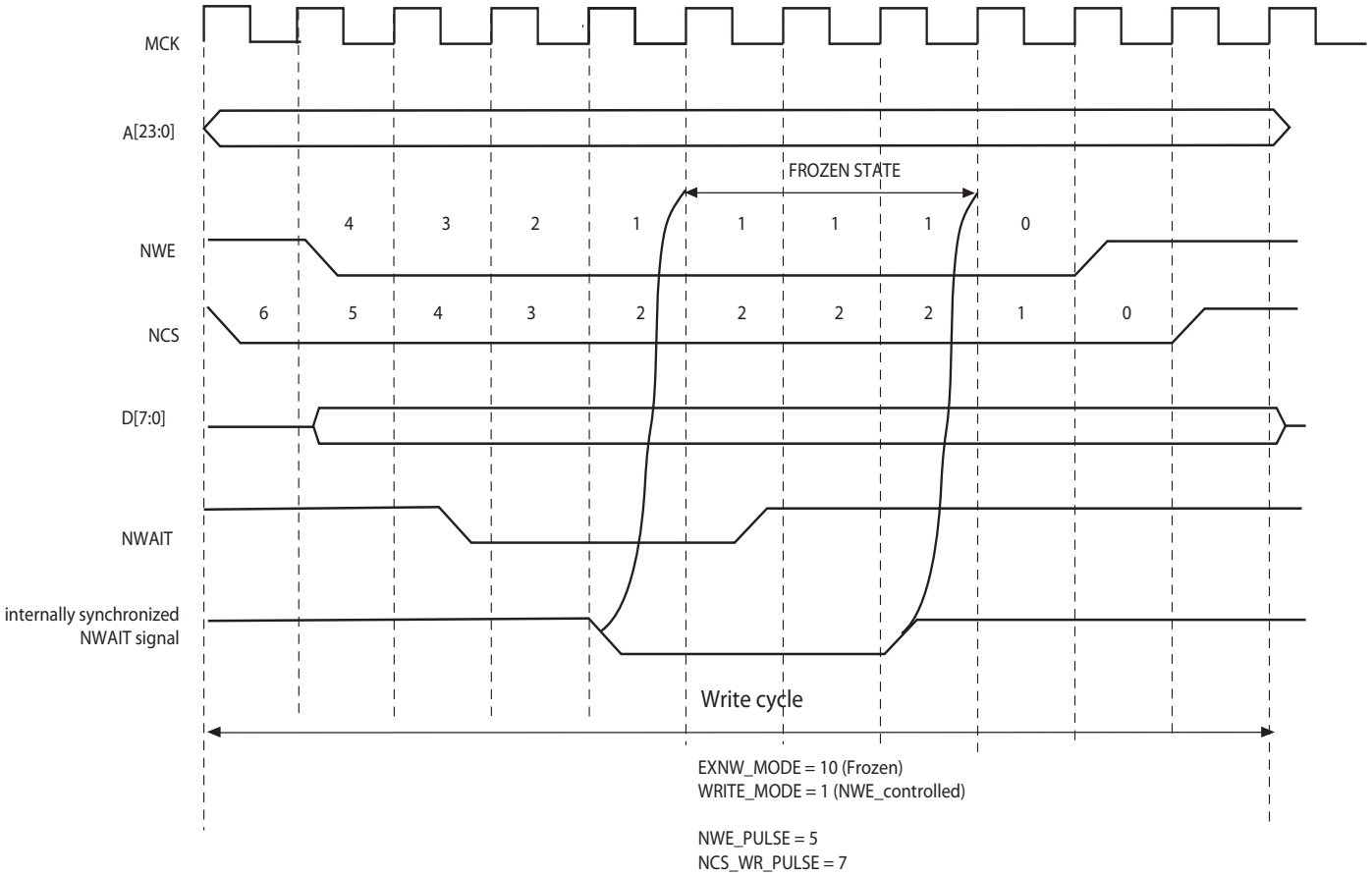
Downloaded from Arrow.com.

34.13.2 Frozen Mode

When the external device asserts the NWAIT signal (active low), and after internal synchronization of this signal, the SMC state is frozen, i.e., SMC internal counters are frozen, and all control signals remain unchanged. When the resynchronized NWAIT signal is deasserted, the SMC completes the access, resuming the access from the point where it was stopped. See Figure 34-25. This mode must be selected when the external device uses the NWAIT signal to delay the access and to freeze the SMC.

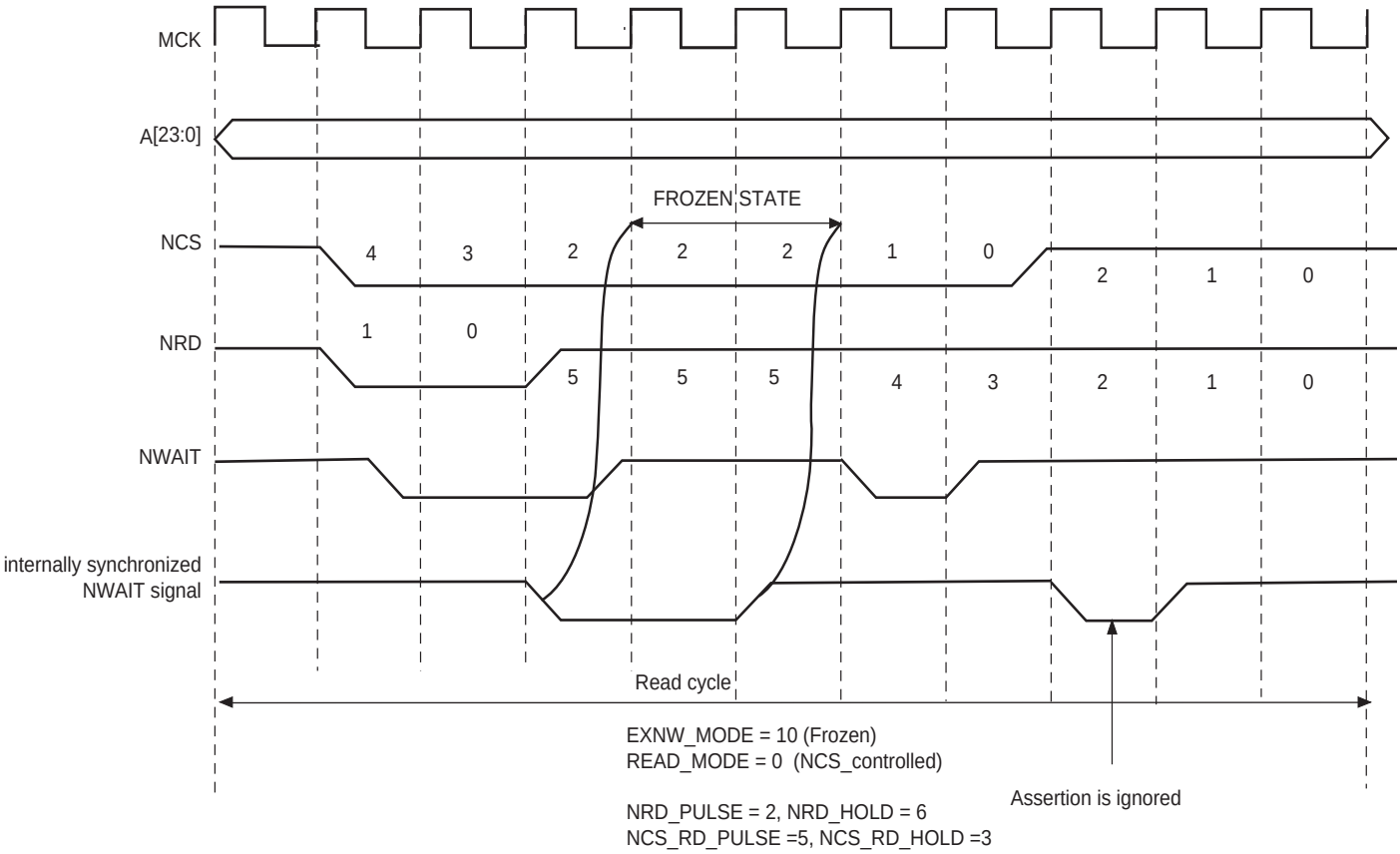
The assertion of the NWAIT signal outside the expected period is ignored as illustrated in Figure 34-26.

Figure 34-25. Write Access with NWAIT Assertion in Frozen Mode (SMC_MODE.EXNW_MODE = 10)



Downloaded from Arrow.com.

Figure 34-26. Read Access with NWAIT Assertion in Frozen Mode (SMC_MODE.EXNW_MODE = 10)



Downloaded from Arrow.com.

34.13.3 Ready Mode

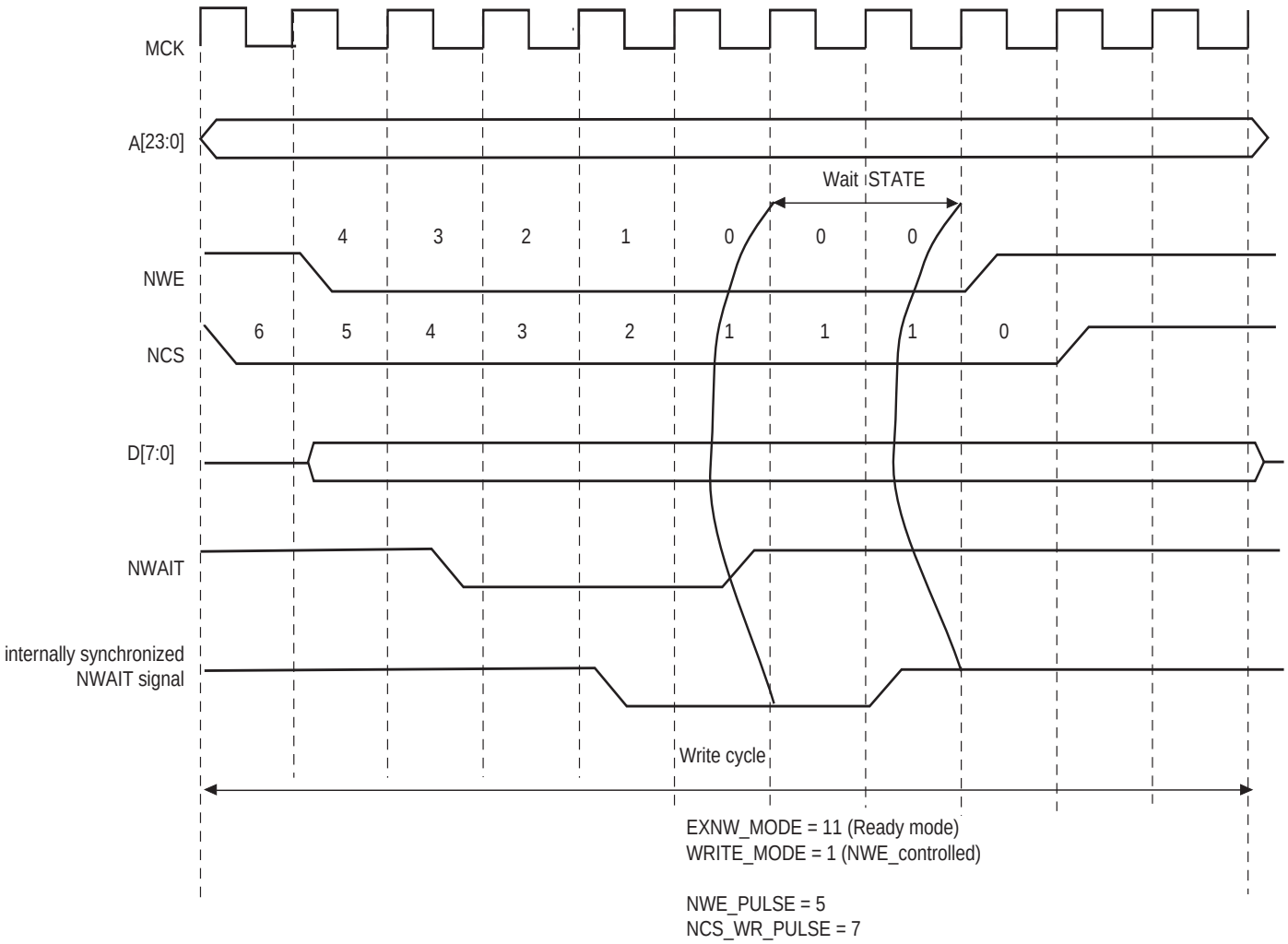
In Ready mode (SMC_MODE.EXNW_MODE = 11), the SMC behaves differently. Normally, the SMC begins the access by down counting the setup and pulse counters of the read/write controlling signal. In the last cycle of the pulse phase, the resynchronized NWAIT signal is examined.

If asserted, the SMC suspends the access as shown in Figure 34-27 and Figure 34-28. After deassertion, the access is completed: the hold step of the access is performed.

This mode must be selected when the external device uses deassertion of the NWAIT signal to indicate its ability to complete the read or write operation.

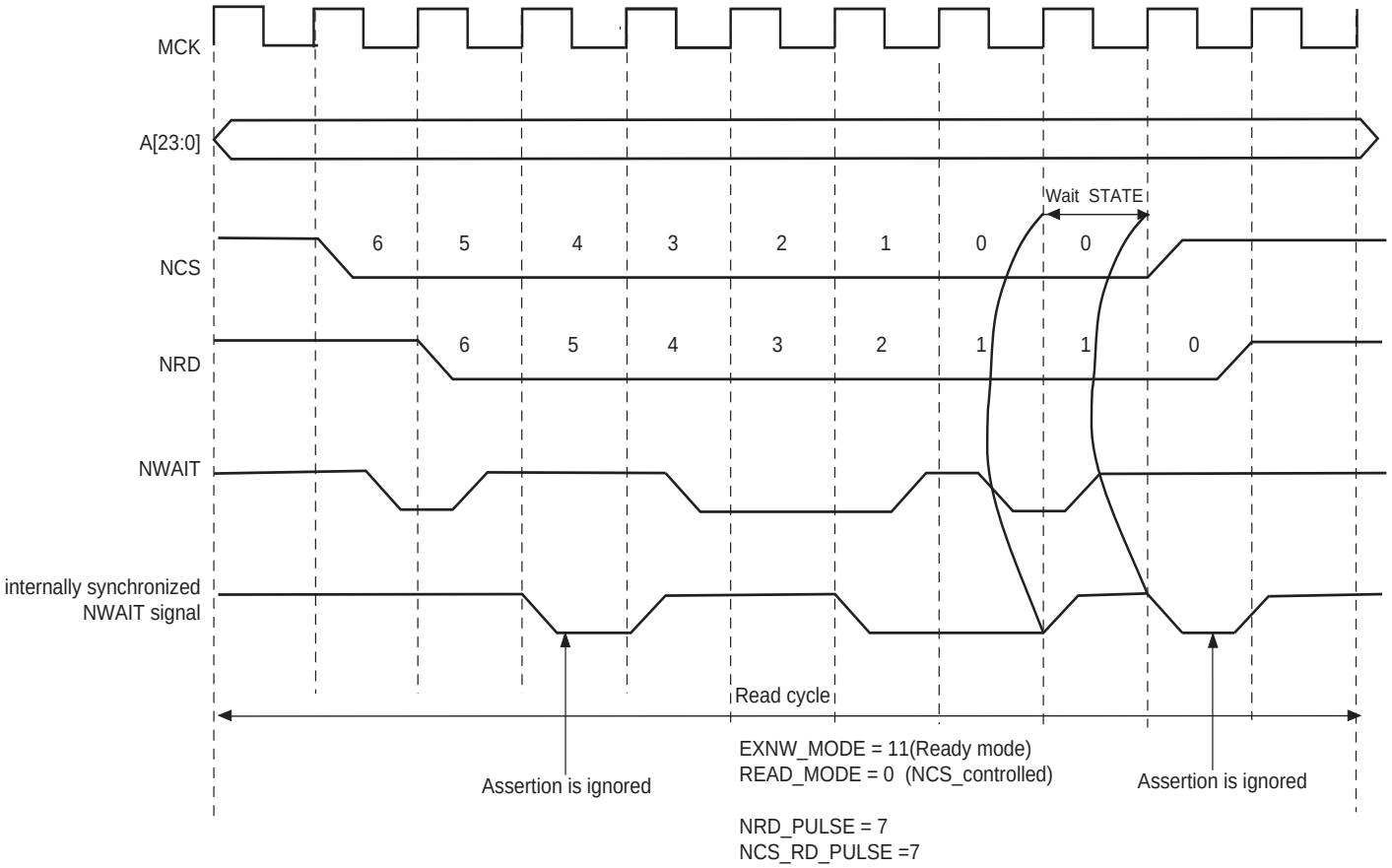
If the NWAIT signal is deasserted before the end of the pulse, or asserted after the end of the pulse of the controlling read/write signal, it has no impact on the access length as shown in Figure 34-28.

Figure 34-27. NWAIT Assertion in Write Access: Ready Mode (SMC_MODE.EXNW_MODE = 11)



Downloaded from Arrow.com.

Figure 34-28. NWAIT Assertion in Read Access: Ready Mode (SMC_MODE.EXNW_MODE = 11)



Downloaded from Arrow.com.

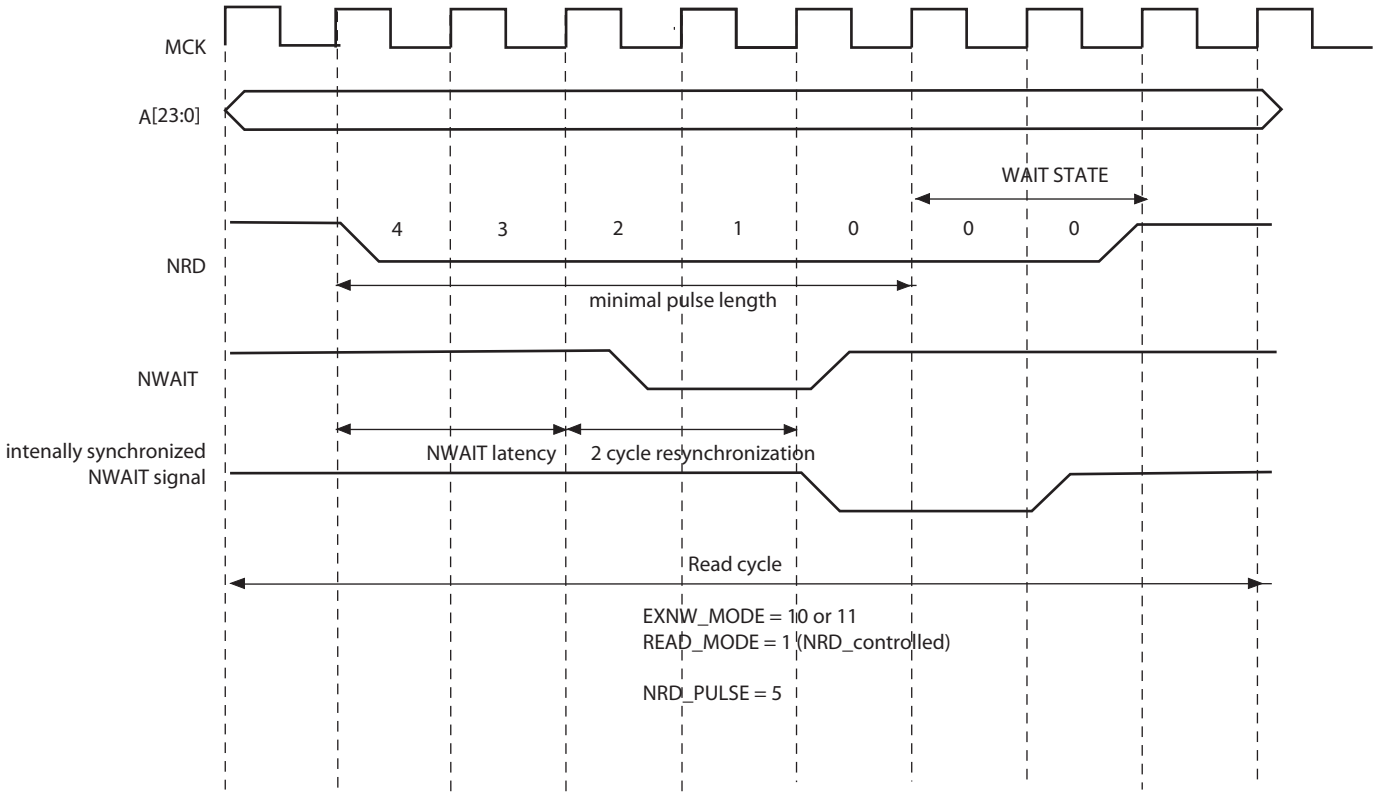
34.13.4 NWAIT Latency and Read/Write Timings

There may be a latency between the assertion of the read/write controlling signal and the assertion of the NWAIT signal by the device. The programmed pulse length of the read/write controlling signal must be at least equal to this latency plus the 2 cycles of resynchronization + one cycle. Otherwise, the SMC may enter the hold state of the access without detecting the NWAIT signal assertion. This is true in Frozen mode as well as in Ready mode. This is illustrated on Figure 34-29.

When SMC_MODE.EXNW_MODE is enabled (ready or frozen), the user must program a pulse length of the read and write controlling signal of at least:

Minimal pulse length = NWAIT latency + 2 resynchronization cycles + 1 cycle

Figure 34-29. NWAIT Latency



Downloaded from Arrow.com.

34.14 Slow Clock Mode

The SMC is able to automatically apply a set of “Slow clock mode” read/write waveforms when an internal signal driven by the Power Management Controller is asserted because MCK has been turned to a very slow clock rate (typically 32kHz clock rate). In this mode, the user-programmed waveforms are ignored and the Slow clock mode waveforms are applied. This mode is provided so as to avoid reprogramming the User Interface with appropriate waveforms at a very slow clock rate. When activated, the Slow clock mode is active on all chip selects.

34.14.1 Slow Clock Mode Waveforms

Figure 34-30 illustrates the read and write operations in Slow clock mode. They are valid on all chip selects. Table 34-7 indicates the value of read and write parameters in Slow clock mode.

Figure 34-30. Read/Write Cycles in Slow Clock Mode

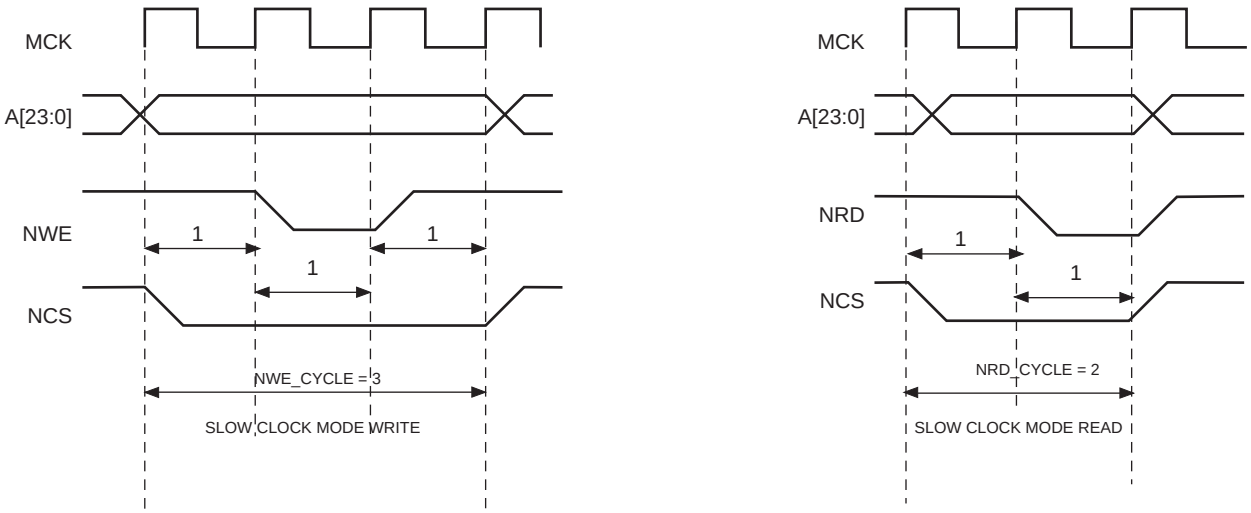


Table 34-7. Read and Write Timing Parameters in Slow Clock Mode

Read Parameters	Duration (cycles)	Write Parameters	Duration (cycles)
NRD_SETUP	1	NWE_SETUP	1
NRD_PULSE	1	NWE_PULSE	1
NCS_RD_SETUP	0	NCS_WR_SETUP	0
NCS_RD_PULSE	2	NCS_WR_PULSE	3
NRD_CYCLE	2	NWE_CYCLE	3

Downloaded from Arrow.com.

34.14.2 Switching from (to) Slow Clock Mode to (from) Normal Mode

When switching from Slow clock mode to Normal mode, the current Slow clock mode transfer is completed at a high clock rate, with the set of Slow clock mode parameters.See Figure 34-31. The external device may not be fast enough to support such timings.

Figure 34-32 illustrates the recommended procedure to switch from one mode to the other.

Figure 34-31. Clock Rate Transition Occurs while the SMC is Performing a Write Operation

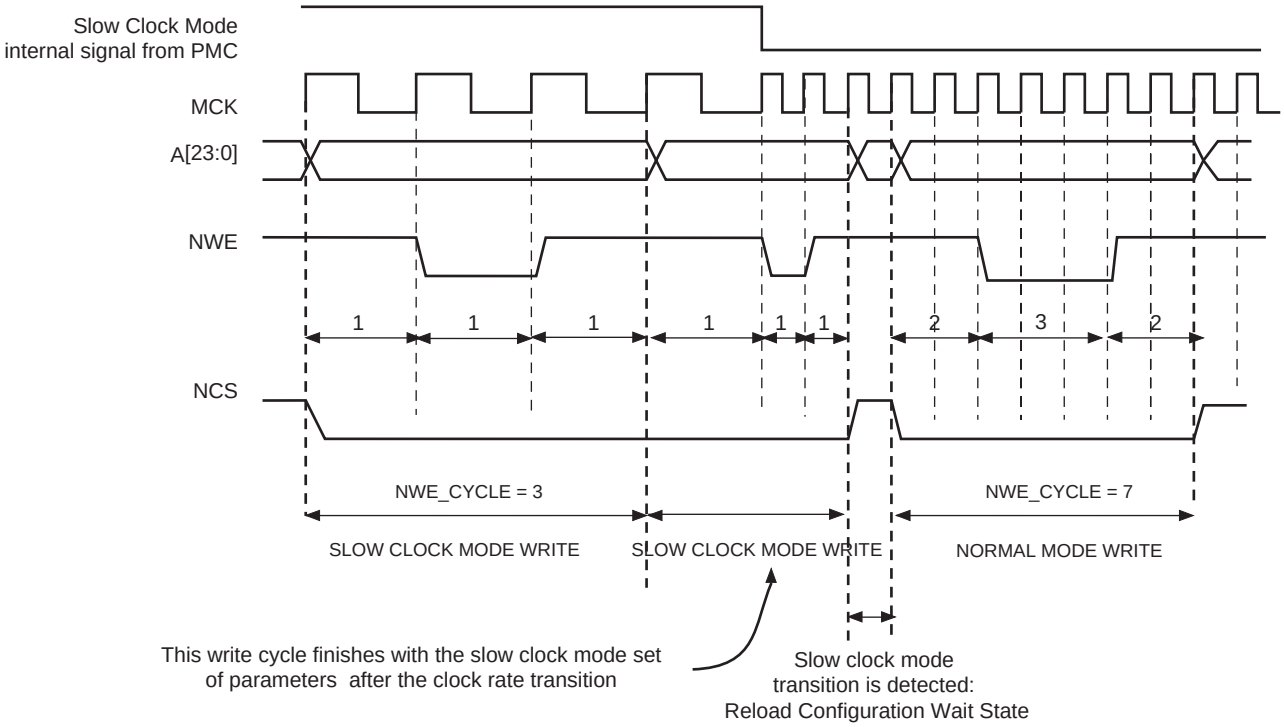
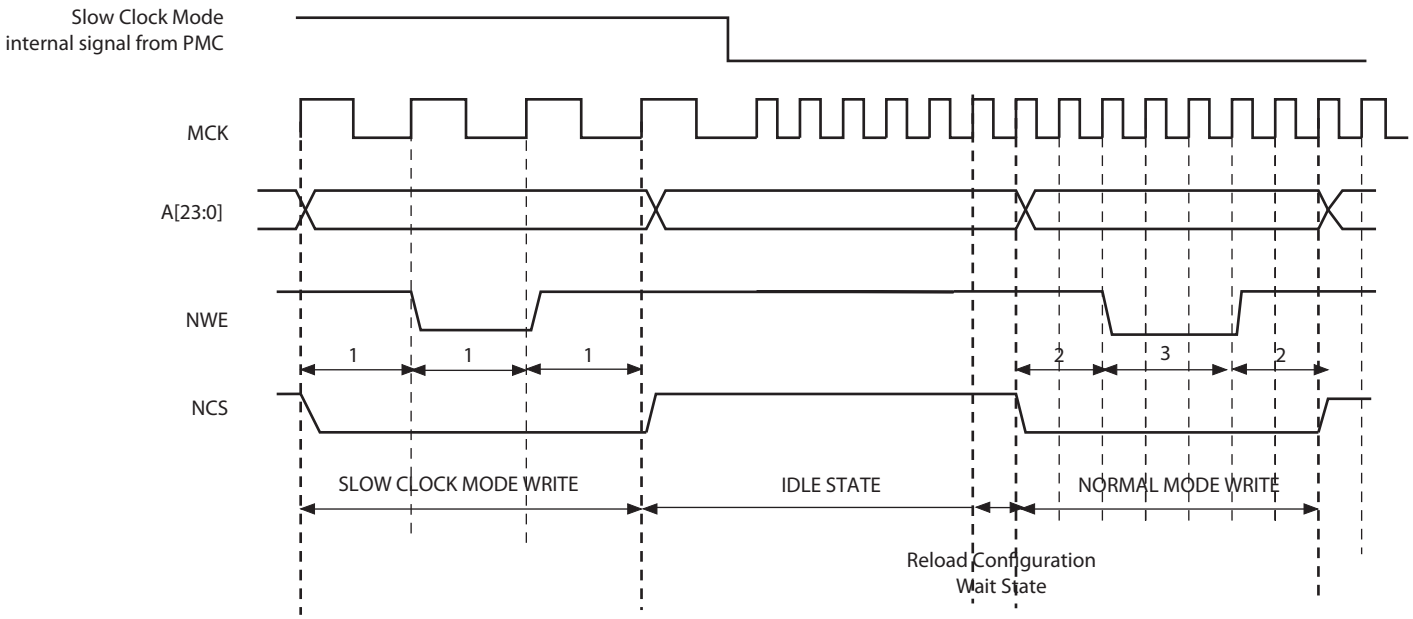


Figure 34-32. Recommended Procedure to Switch from Slow Clock Mode to Normal Mode or from Normal Mode to Slow Clock Mode



Downloaded from Arrow.com.

34.15 Asynchronous Page Mode

The SMC supports asynchronous burst reads in Page mode, provided that the Page mode is enabled (SMC_MODE.PMEN =1). The page size must be configured in the SMC_MODE register (PS field) to 4, 8, 16 or 32 bytes.

The page defines a set of consecutive bytes into memory. A 4-byte page (resp. 8-, 16-, 32-byte page) is always aligned to 4-byte boundaries (resp. 8-, 16-, 32-byte boundaries) of memory. The MSB of data address defines the address of the page in memory, the LSB of address define the address of the data in the page as detailed in Table 34-8.

With Page mode memory devices, the first access to one page (t_{pa}) takes longer than the subsequent accesses to the page (t_{sa}) as shown in Figure 34-33. When in Page mode, the SMC enables the user to define different read timings for the first access within one page, and next accesses within the page.

Table 34-8. Page Address and Data Address within a Page

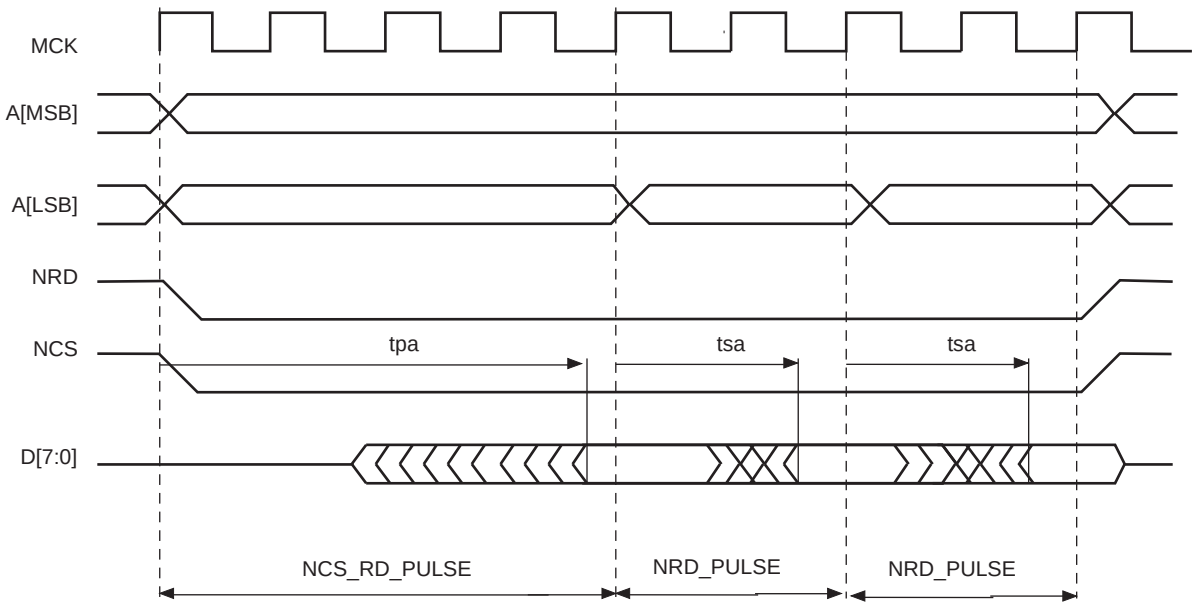
Page Size	Page Address(1)	Data Address in the Page
4 bytes	A[23:2]	A[1:0]
8 bytes	A[23:3]	A[2:0]
16 bytes	A[23:4]	A[3:0]
32 bytes	A[23:5]	A[4:0]

Note: 1. “A” denotes the address bus of the memory device.

34.15.1 Protocol and Timings in Page Mode

Figure 34-33 shows the NRD and NCS timings in Page mode access.

Figure 34-33. Page Mode Read Protocol (Address MSB and LSB are defined in Table 34-8)



The NRD and NCS signals are held low during all read transfers, whatever the programmed values of the setup and hold timings in the User Interface may be. Moreover, the NRD and NCS timings are identical. The pulse length of the first access to the page is defined with the NCS_RD_PULSE field of the SMC_PULSE register. The pulse length of subsequent accesses within the page are defined using the NRD_PULSE parameter.

Downloaded from Arrow.com.

In Page mode, the programming of the read timings is described in Table 34-9:

Table 34-9. Programming of Read Timings in Page Mode

Parameter	Value	Definition
READ_MODE	'x'	No impact
NCS_RD_SETUP	'x'	No impact
NCS_RD_PULSE	t_{pa}	Access time of first access to the page
NRD_SETUP	'x'	No impact
NRD_PULSE	t_{sa}	Access time of subsequent accesses in the page
NRD_CYCLE	'x'	No impact

The SMC does not check the coherency of timings. It will always apply the NCS_RD_PULSE timings as page access timing (t_{pa}) and the NRD_PULSE for accesses to the page (t_{sa}), even if the programmed value for t_{pa} is shorter than the programmed value for t_{sa}.

34.15.2 Page Mode Restriction

The Page mode is not compatible with the use of the NWAIT signal. Using the Page mode and the NWAIT signal may lead to unpredictable behavior.

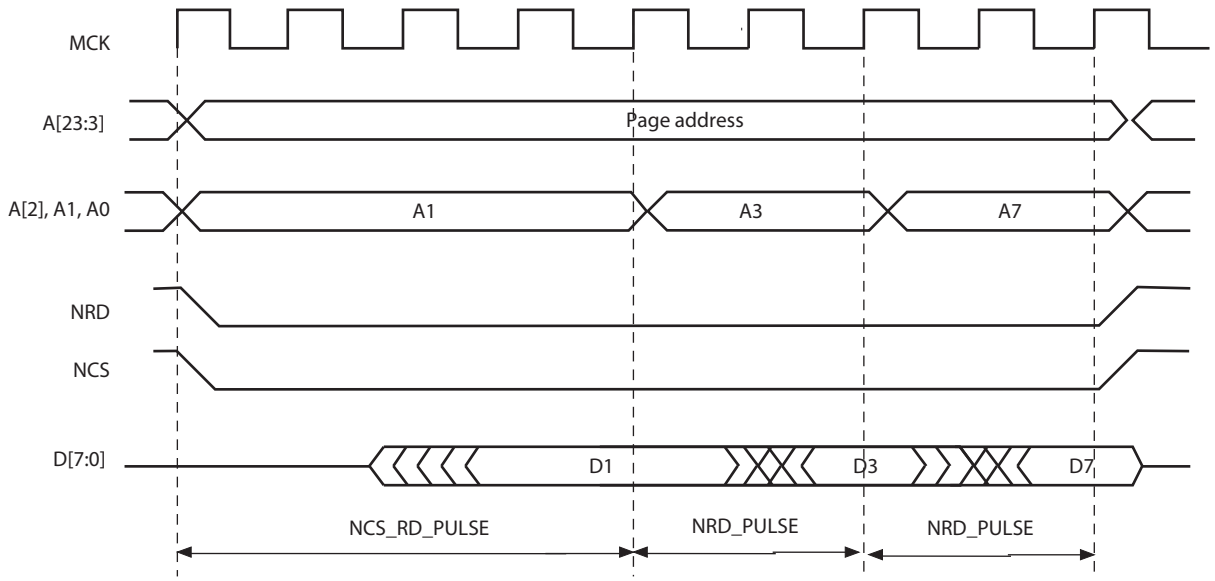
34.15.3 Sequential and Non-sequential Accesses

If the chip select and the MSB of addresses as defined in Table 34-8 are identical, then the current access lies in the same page as the previous one, and no page break occurs.

Using this information, all data within the same page, sequential or not sequential, are accessed with a minimum access time (t_{sa}). Figure 34-34 illustrates access to an 8-bit memory device in Page mode, with 8-byte pages. Access to D1 causes a page access with a long access time (t_{pa}). Accesses to D3 and D7, though they are not sequential accesses, only require a short access time (t_{sa}).

If the MSB of addresses are different, the SMC performs the access of a new page. In the same way, if the chip select is different from the previous access, a page break occurs. If two sequential accesses are made to the Page mode memory, but separated by an other internal or external peripheral access, a page break occurs on the second access because the chip select of the device was deasserted between both accesses.

Figure 34-34. Access to Non-Sequential Data within the Same Page



Downloaded from Arrow.com.

34.16 Static Memory Controller (SMC) User Interface

The SMC is programmed using the registers listed in Table 34-10. For each chip select, a set of four registers is used to program the parameters of the external device connected on it. In Table 34-10, “CS_number” denotes the chip select number. 16 bytes (0x10) are required per chip select.

Note: The user must confirm the SMC configuration by writing any one of the SMC_MODE registers.

Table 34-10. Register Mapping

Offset	Register	Name	Access	Reset
0x10 x CS_number + 0x00	SMC Setup Register	SMC_SETUP	Read/Write	0x01010101
0x10 x CS_number + 0x04	SMC Pulse Register	SMC_PULSE	Read/write	0x01010101
0x10 x CS_number + 0x08	SMC Cycle Register	SMC_CYCLE	Read/Write	0x00030003
0x10 x CS_number + 0x0C	SMC Mode Register	SMC_MODE	Read/Write	0x10001003
0x80	SMC Off-Chip Memory Scrambling Register	SMC_OCMS	Read/Write	0x00000000
0x84	SMC Off-Chip Memory Scrambling KEY1 Register	SMC_KEY1	Write-once	0x00000000
0x88	SMC Off-Chip Memory Scrambling KEY2 Register	SMC_KEY2	Write-once	0x00000000
0xE4	SMC Write Protection Mode Register	SMC_WPMR	Read/Write	0x00000000
0xE8	SMC Write Protection Status Register	SMC_WPSR	Read-only	0x00000000
0xEC-0xFC	Reserved	–	–	–

Notes: 1. All unlisted offset values are considered as ‘reserved’.

Downloaded from Arrow.com.

34.16.1 SMC Setup Register

Name: SMC_SETUP[0..3]

Address: 0x40080000 [0], 0x40080010 [1], 0x40080020 [2], 0x40080030 [3]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	NCS_RD_SETUP					

23	22	21	20	19	18	17	16
–	–	NRD_SETUP					

15	14	13	12	11	10	9	8
–	–	NCS_WR_SETUP					

7	6	5	4	3	2	1	0
–	–	NWE_SETUP					

This register can only be written if the WPEN bit is cleared in the “SMC Write Protection Mode Register” .

- **NWE_SETUP: NWE Setup Length**

The NWE signal setup length is defined as:

NWE setup length = (128* NWE_SETUP[5] + NWE_SETUP[4:0]) clock cycles

- **NCS_WR_SETUP: NCS Setup Length in WRITE Access**

In write access, the NCS signal setup length is defined as:

NCS setup length = (128* NCS_WR_SETUP[5] + NCS_WR_SETUP[4:0]) clock cycles

- **NRD_SETUP: NRD Setup Length**

The NRD signal setup length is defined in clock cycles as:

NRD setup length = (128* NRD_SETUP[5] + NRD_SETUP[4:0]) clock cycles

- **NCS_RD_SETUP: NCS Setup Length in READ Access**

In read access, the NCS signal setup length is defined as:

NCS setup length = (128* NCS_RD_SETUP[5] + NCS_RD_SETUP[4:0]) clock cycles

Downloaded from Arrow.com.

34.16.2 SMC Pulse Register

Name: SMC_PULSE[0..3]

Address: 0x40080004 [0], 0x40080014 [1], 0x40080024 [2], 0x40080034 [3]

Access: Read/Write

31	30	29	28	27	26	25	24
–	NCS_RD_PULSE						

23	22	21	20	19	18	17	16
–	NRD_PULSE						

15	14	13	12	11	10	9	8
–	NCS_WR_PULSE						

7	6	5	4	3	2	1	0
–	NWE_PULSE						

This register can only be written if the WPEN bit is cleared in the “SMC Write Protection Mode Register” .

- **NWE_PULSE: NWE Pulse Length**

The NWE signal pulse length is defined as:

NWE pulse length = (256* NWE_PULSE[6] + NWE_PULSE[5:0]) clock cycles

The NWE pulse length must be at least 1 clock cycle.

- **NCS_WR_PULSE: NCS Pulse Length in WRITE Access**

In write access, the NCS signal pulse length is defined as:

NCS pulse length = (256* NCS_WR_PULSE[6] + NCS_WR_PULSE[5:0]) clock cycles

The NCS pulse length must be at least 1 clock cycle.

- **NRD_PULSE: NRD Pulse Length**

In standard read access, the NRD signal pulse length is defined in clock cycles as:

NRD pulse length = (256* NRD_PULSE[6] + NRD_PULSE[5:0]) clock cycles

The NRD pulse length must be at least 1 clock cycle.

In Page mode read access, the NRD_PULSE parameter defines the duration of the subsequent accesses in the page.

- **NCS_RD_PULSE: NCS Pulse Length in READ Access**

In standard read access, the NCS signal pulse length is defined as:

NCS pulse length = (256* NCS_RD_PULSE[6] + NCS_RD_PULSE[5:0]) clock cycles

The NCS pulse length must be at least 1 clock cycle.

In Page mode read access, the NCS_RD_PULSE parameter defines the duration of the first access to one page.

Downloaded from Arrow.com.

34.16.3 SMC Cycle Register

Name: SMC_CYCLE[0..3]

Address: 0x40080008 [0], 0x40080018 [1], 0x40080028 [2], 0x40080038 [3]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	NRD_CYCLE

23	22	21	20	19	18	17	16
NRD_CYCLE							

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	NWE_CYCLE

7	6	5	4	3	2	1	0
NWE_CYCLE							

This register can only be written if the WPEN bit is cleared in the “SMC Write Protection Mode Register” .

- **NWE_CYCLE: Total Write Cycle Length**

The total write cycle length is the total duration in clock cycles of the write cycle. It is equal to the sum of the setup, pulse and hold steps of the NWE and NCS signals. It is defined as:

Write cycle length = (NWE_CYCLE[8:7]*256 + NWE_CYCLE[6:0]) clock cycles

- **NRD_CYCLE: Total Read Cycle Length**

The total read cycle length is the total duration in clock cycles of the read cycle. It is equal to the sum of the setup, pulse and hold steps of the NRD and NCS signals. It is defined as:

Read cycle length = (NRD_CYCLE[8:7]*256 + NRD_CYCLE[6:0]) clock cycles

Downloaded from Arrow.com.

34.16.4 SMC Mode Register

Name: SMC_MODE[0..3]

Address: 0x4008000C [0], 0x4008001C [1], 0x4008002C [2], 0x4008003C [3]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	PS		–	–	–	PMEN
23	22	21	20	19	18	17	16
–	–	–	TDF_MODE	TDF_CYCLES			
15	14	13	12	11	10	9	8
–	–	–	DBW	–	–	–	BAT
7	6	5	4	3	2	1	0
–	–	EXNW_MODE		–	–	WRITE_MODE	READ_MODE

This register can only be written if the WPEN bit is cleared in the “SMC Write Protection Mode Register” .

The user must confirm the SMC configuration by writing any one of the SMC_MODE registers.

- **READ_MODE: Read Mode**

0: The read operation is controlled by the NCS signal.

– If TDF cycles are programmed, the external bus is marked busy after the rising edge of NCS.

– If TDF optimization is enabled (TDF_MODE =1), TDF wait states are inserted after the setup of NCS.

1: The read operation is controlled by the NRD signal.

– If TDF cycles are programmed, the external bus is marked busy after the rising edge of NRD.

– If TDF optimization is enabled (TDF_MODE =1), TDF wait states are inserted after the setup of NRD.

- **WRITE_MODE: Write Mode**

0: The write operation is controlled by the NCS signal.

– If TDF optimization is enabled (TDF_MODE =1), TDF wait states will be inserted after the setup of NCS.

1: The write operation is controlled by the NWE signal.

– If TDF optimization is enabled (TDF_MODE =1), TDF wait states will be inserted after the setup of NWE.

- **EXNW_MODE: NWAIT Mode**

The NWAIT signal is used to extend the current read or write signal. It is only taken into account during the pulse phase of the read and write controlling signal. When the use of NWAIT is enabled, at least one cycle hold duration must be programmed for the read and write controlling signal.

Value	Name	Description
0	DISABLED	Disabled–The NWAIT input signal is ignored on the corresponding chip select.
1	–	Reserved
2	FROZEN	Frozen Mode–If asserted, the NWAIT signal freezes the current read or write cycle. After deassertion, the read/write cycle is resumed from the point where it was stopped.
3	READY	Ready Mode–The NWAIT signal indicates the availability of the external device at the end of the pulse of the controlling read or write signal, to complete the access. If high, the access normally completes. If low, the access is extended until NWAIT returns high.

Downloaded from Arrow.com.

- **BAT: Byte Access Type**

This field is used only if DBW defines a 16-bit data bus.

Value	Name	Description
0	BYTE_SELECT	Byte select access type: - Write operation is controlled using NCS, NWE, NBS0, NBS1. - Read operation is controlled using NCS, NRD, NBS0, NBS1.
1	BYTE_WRITE	Byte write access type: - Write operation is controlled using NCS, NWR0, NWR1. - Read operation is controlled using NCS and NRD.

- **DBW: Data Bus Width**

Value	Name	Description
0	8_BIT	8-bit Data Bus
1	16_BIT	16-bit Data Bus

- **TDF_CYCLES: Data Float Time**

This field gives the integer number of clock cycles required by the external device to release the data after the rising edge of the read controlling signal. The SMC always provide one full cycle of bus turnaround after the TDF_CYCLES period. The external bus cannot be used by another chip select during TDF_CYCLES + 1 cycles. From 0 up to 15 TDF_CYCLES can be set.

- **TDF_MODE: TDF Optimization**

0: TDF optimization disabled–the number of TDF wait states is inserted before the next access begins.

1: TDF optimization enabled–the number of TDF wait states is optimized using the setup period of the next read/write access.

- **PMEN: Page Mode Enabled**

0: Standard read is applied.

1: Asynchronous burst read in page mode is applied on the corresponding chip select.

- **PS: Page Size**

If page mode is enabled, this field indicates the size of the page in bytes.

Value	Name	Description
0	4_BYTE	4-byte page
1	8_BYTE	8-byte page
2	16_BYTE	16-byte page
3	32_BYTE	32-byte page

Downloaded from Arrow.com.

34.16.5 SMC Off-Chip Memory Scrambling Register

Name: SMC_OCMS

Address: 0x40080080

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	CS3SE	CS2SE	CS1SE	CS0SE
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	SMSE

- **CSxSE: Chip Select (x = 0 to 3) Scrambling Enable**

0: Disable scrambling for CSx.

1: Enable scrambling for CSx.

- **SMSE: Static Memory Controller Scrambling Enable**

0: Disable scrambling for SMC access.

1: Enable scrambling for SMC access.

Downloaded from Arrow.com.

34.16.6 SMC Off-Chip Memory Scrambling Key1 Register

Name: SMC_KEY1

Address: 0x40080084

Access: Write-once[1]

31	30	29	28	27	26	25	24
KEY1							

23	22	21	20	19	18	17	16
KEY1							

15	14	13	12	11	10	9	8
KEY1							

7	6	5	4	3	2	1	0
KEY1							

Note: 1. 'Write-once' access indicates that the first write access after a system reset prevents any further modification of the value of this register.

- **KEY1: Off-Chip Memory Scrambling (OCMS) Key Part 1**

When off-chip memory scrambling is enabled, KEY1 and KEY2 values determine data scrambling.

Downloaded from Arrow.com.

34.16.7 SMC Off-Chip Memory Scrambling Key2 Register

Name: SMC_KEY2

Address: 0x40080088

Access: Write-once[1]

31	30	29	28	27	26	25	24
KEY2							

23	22	21	20	19	18	17	16
KEY2							

15	14	13	12	11	10	9	8
KEY2							

7	6	5	4	3	2	1	0
KEY2							

Notes: 1. 'Write-once' access indicates that the first write access after a system reset prevents any further modification of the value of this register.

- **KEY2: Off-Chip Memory Scrambling (OCMS) Key Part 2**

When off-chip memory scrambling is enabled, KEY1 and KEY2 values determine data scrambling.

Downloaded from Arrow.com.

34.16.8 SMC Write Protection Mode Register

Name: SMC_WPMR

Address: 0x400800E4

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							

23	22	21	20	19	18	17	16
WPKEY							

15	14	13	12	11	10	9	8
WPKEY							

7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	WPEN

- **WPEN: Write Protect Enable**

0: Disables the write protection if WPKEY corresponds to 0x534D43 (“SMC” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x534D43 (“SMC” in ASCII).

See Section 34.9.5 ”Register Write Protection” for the list of registers that can be write-protected.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x534D43	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

Downloaded from Arrow.com.

34.16.9 SMC Write Protection Status Register

Name: SMC_WPSR

Address: 0x400800E8

Type: Read-only

31	30	29	28	27	26	25	24
—	—	—	—	—	—	—	—

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	WPVS

- **WPVS: Write Protection Violation Status**

0: No write protection violation has occurred since the last read of the SMC_WPSR register.

1: A write protection violation has occurred since the last read of the SMC_WPSR register. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protection Violation Source**

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

Downloaded from Arrow.com.

35. DMA Controller (XDMAC)

35.1 Description

The DMA Controller (XDMAC) is a AHB-protocol central direct memory access controller. It performs peripheral data transfer and memory move operations over one or two bus ports through the unidirectional communication channel. Each channel is fully programmable and provides both peripheral or memory-to-memory transfer. The channel features are configurable at implementation.

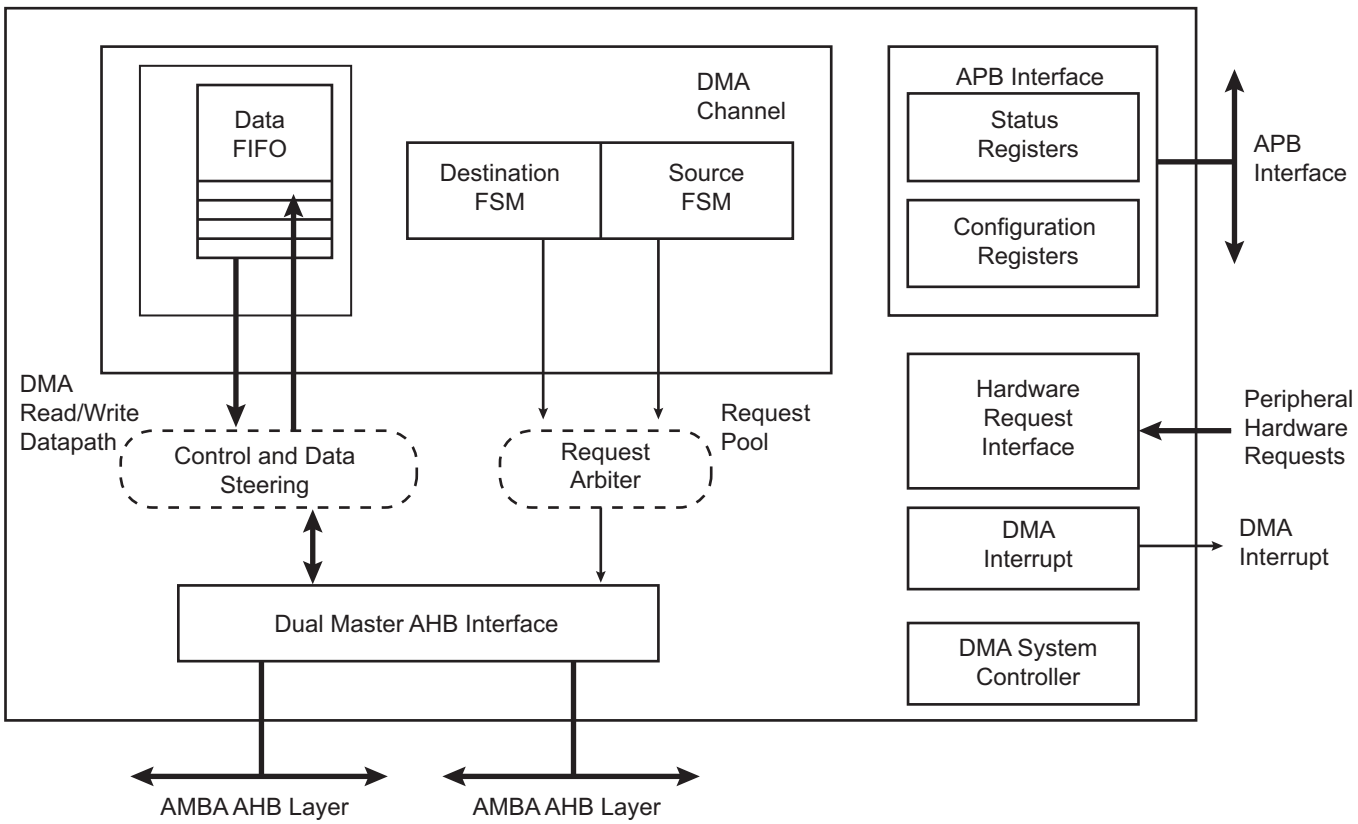
35.2 Embedded Characteristics

- 2 AHB Master Interface
- 24 DMA Channels
- 44 Hardware Requests
- 3.1 Kbytes Embedded FIFO
- Supports Peripheral to Memory, Memory to Peripheral, or Memory to Memory Transfer Operations
- Peripheral DMA Operation Runs on Bytes (8-bit), Half-Word (16-bit) and Word (32-bit)
- Memory DMA Operation Runs on Bytes (8 bit), Half-Word (16-bit) and Word (32 -bit)
- Supports Hardware and Software Initiated Transfers
- Supports Linked List Operations
- Supports Incrementing or Fixed Addressing Mode
- Supports Programmable Independent Data Striding for Source and Destination
- Supports Programmable Independent Microblock Striding for Source and Destination
- Configurable Priority Group and Arbitration Policy
- Programmable AHB Burst Length
- Configuration Interface Accessible through APB Interface
- XDMAC Architecture Includes Multiport FIFO
- Multiple View Channel Descriptor Supported
- Automatic Flush of Channel Trailing Bytes
- Automatic Coarse-Grain and Fine-Grain Clock Gating
- Hardware Acceleration of Memset Pattern

Downloaded from Arrow.com.

35.3 Block Diagram

Figure 35-1. DMA Controller (XDMAC) Block Diagram



Downloaded from Arrow.com.

35.4 DMA Controller Peripheral Connections

The DMA Controller handles the transfer between peripherals and memory and receives requests from the peripherals listed in Table 35-1.

For each listed DMA channel number, the SIF and/or DIF bits in XDMAC_CCx must be programmed with a value compatible with the MATRIX “Master to Slave Access” definition provided in Section 18. “Bus Matrix (MATRIX)”.

Depending on transfer descriptor location, the NDAIF bit in XDMAC_CNDAx must be programmed with a value compatible with the MATRIX “Master to Slave Access” definition provided in Section 18. “Bus Matrix (MATRIX)”.

Table 35-1. Peripheral Hardware Requests

Peripheral Name	Transfer Type	HW Interface Number (XDMAC_CC.PERID)
HSMCI	Transmit/Receive	0
SPI0	Transmit	1
SPI0	Receive	2
SPI1	Transmit	3
SPI1	Receive	4
QSPI	Transmit	5
QSPI	Receive	6
USART0	Transmit	7
USART0	Receive	8
USART1	Transmit	9
USART1	Receive	10
USART2	Transmit	11
USART2	Receive	12
PWM0	Transmit	13
TWIHS0	Transmit	14
TWIHS0	Receive	15
TWIHS1	Transmit	16
TWIHS1	Receive	17
TWIHS2	Transmit	18
TWIHS2	Receive	19
UART0	Transmit	20
UART0	Receive	21
UART1	Transmit	22
UART1	Receive	23
UART2	Transmit	24
UART2	Receive	25
UART3	Transmit	26
UART3	Receive	27
UART4	Transmit	28
UART4	Receive	29
DACC	Transmit	30

Downloaded from Arrow.com.

Table 35-1. Peripheral Hardware Requests (Continued)

Peripheral Name	Transfer Type	HW Interface Number (XDMAC_CC.PERID)
SSC	Transmit	32
SSC	Receive	33
PIOA	Receive	34
AFEC0	Receive	35
AFEC1	Receive	36
AES	Transmit	37
AES	Receive	38
PWM1	Transmit	39
TC0	Receive	40
TC3	Receive	41
TC6	Receive	42
TC9	Receive	43
I2SC0	Transmit Left	44
I2SC0	Receive Left	45
I2SC1	Transmit Left	46
I2SC1	Receive Left	47
I2SC0	Transmit Right	48
I2SC0	Receive Right	49
I2SC1	Transmit Right	50
I2SC1	Receive Right	51

Downloaded from Arrow.com.

35.5 Functional Description

35.5.1 Basic Definitions

Source Peripheral: Slave device, memory mapped on the interconnection network, from where the XDMAC reads data. The source peripheral teams up with a destination peripheral to form a channel. A data read operation is scheduled when the peripheral transfer request is asserted.

Destination Peripheral: Slave device, memory mapped on the interconnection network, to which the XDMAC writes. A write data operation is scheduled when the peripheral transfer request is asserted.

Channel: The data movement between source and destination creates a logical channel.

Transfer Type: The transfer is hardware synchronized when it is paced by the peripheral hardware request, otherwise the transfer is self-triggered (memory to memory transfer).

35.5.2 Transfer Hierarchy Diagram

XDMAC Master Transfer: The Master Transfer is composed of a linked list of blocks. The channel address, control and configuration registers can be modified at the inter block boundary. The descriptor structure modifies the channel registers conditionally. Interrupts can be generated on a per block basis or when the end of linked list event occurs.

XDMAC Block: An XDMAC block is composed of a programmable number of microblocks. The channel configuration registers remain unchanged at the inter microblock boundary. The source and destination addresses are conditionally updated with a programmable signed number.

XDMAC Microblock: The microblock is composed of a programmable number of data. The channel configuration registers remain unchanged at the data boundary. The data address may be fixed (a FIFO location, a peripheral transmit or receive register), incrementing (a memory mapped area) by a programmable signed number.

XDMAC Burst and Incomplete Burst: In order to improve the overall performance when accessing dynamic external memory, burst access is mandatory. Each data of the microblock is considered as a part of a memory burst. The programmable burst value indicates the largest memory burst allowed on a per channel basis. When the microblock length is not an integral multiple of the burst size, an incomplete burst is performed to read or write the last trailing bytes.

XDMAC Chunk and Incomplete Chunk: When a peripheral synchronized transfer is activated, the microblock splits into a number of data chunks. The chunk size is programmable. The larger the chunk is, the better the performance is. When the transfer size is not a multiple of the chunk size, the last chunk may be incomplete.

Downloaded from Arrow.com.

Figure 35-2. XDMAC Memory Transfer Hierarchy

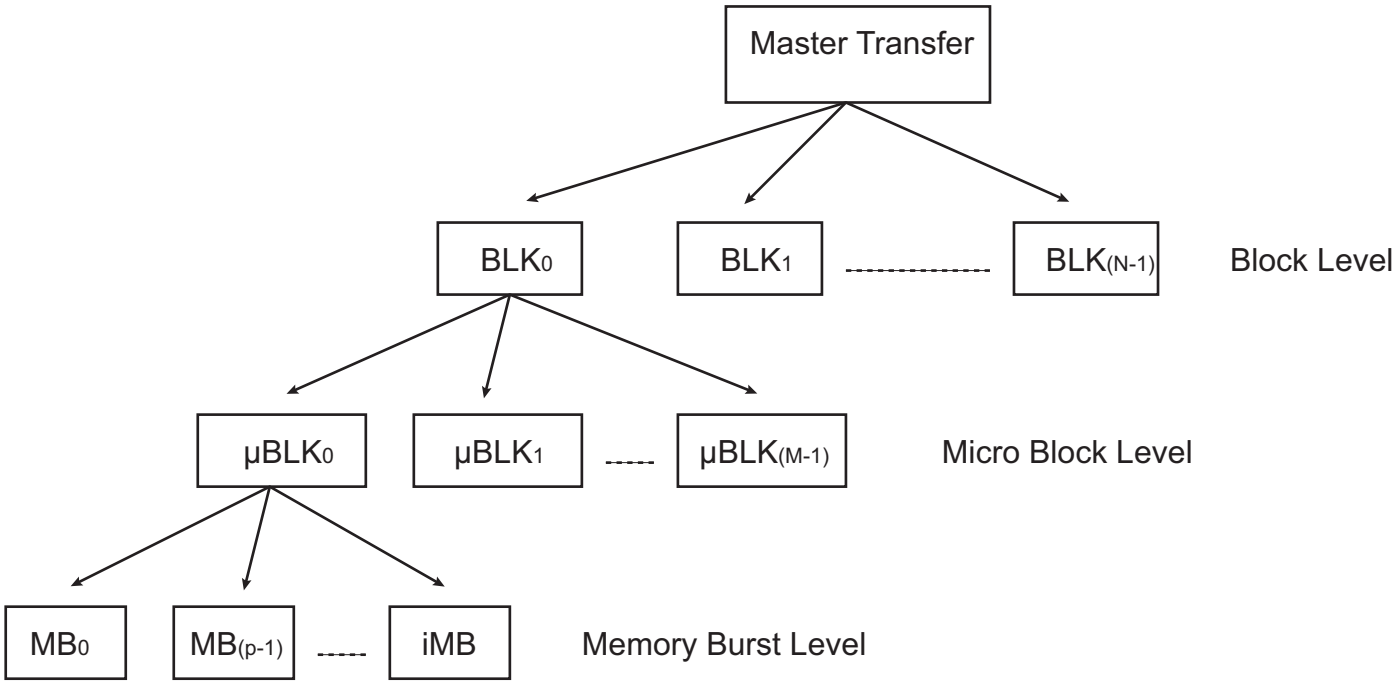
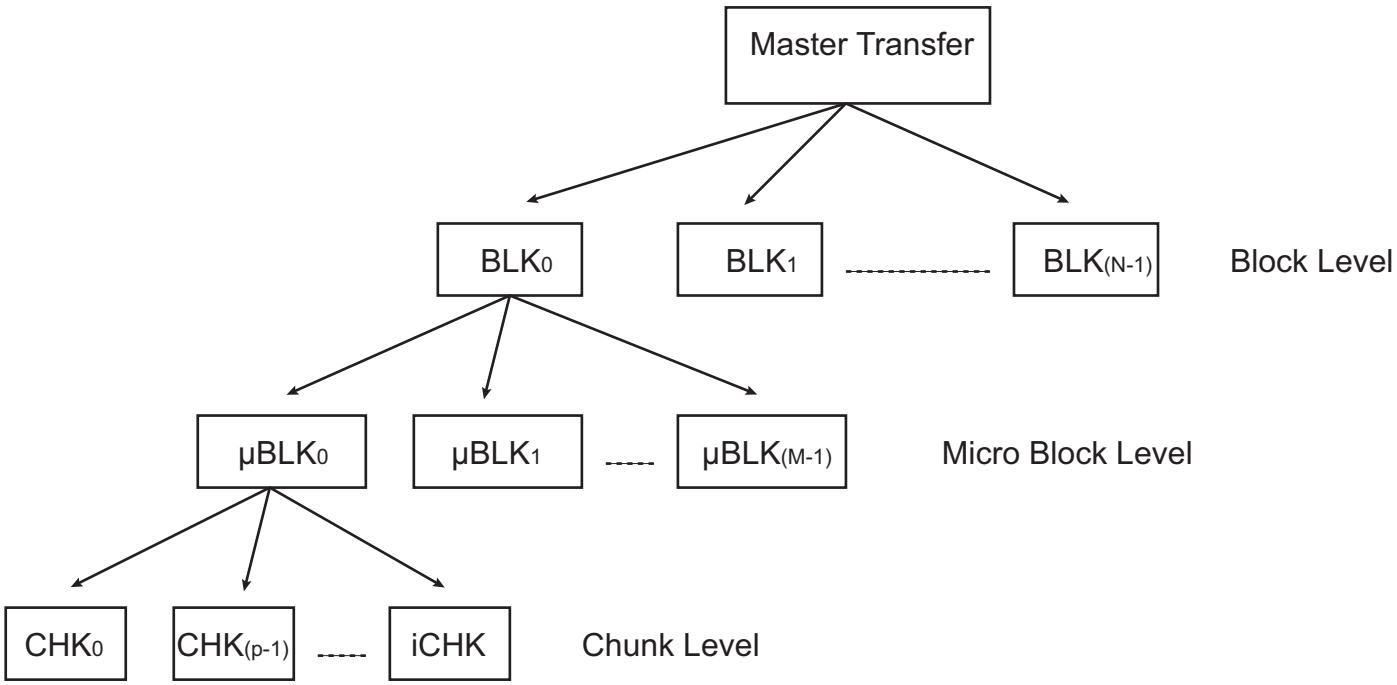


Figure 35-3. XDAMC Peripheral Transfer Hierarchy



Downloaded from Arrow.com.

35.5.3 Peripheral Synchronized Transfer

A peripheral hardware request interface is used to control the pace of the chunk transfer. When a peripheral is ready to transmit or receive a chunk of data, it asserts its request line and the DMA Controller transfers a data to or from the memory to the peripheral.

35.5.3.1 Software Triggered Synchronized Transfer

The Peripheral hardware request can be software controlled using the SWREQ field of the XDMAC Global Channel Software Request Register (XDMAC_GSWR). The peripheral synchronized transfer is paced using a processor write access in the XDMAC_GSWR. Each bit of that register triggers a transfer request. The XDMAC Global Channel Software Request Status Register (XDMAC_GSWS) indicates the status of the request; when set, the request is still pending.

35.5.4 XDMAC Transfer Software Operation

35.5.4.1 Single Block With Single Microblock Transfer

1. Read the XDMAC Global Channel Status Register (XDMAC_GS) to choose a free channel.
2. Clear the pending Interrupt Status bit(s) by reading the chosen XDMAC Channel x Interrupt Status Register (XDMAC_CISx).
3. Write the XDMAC Channel x Source Address Register (XDMAC_CSAx) for channel x.
4. Write the XDMAC Channel x Destination Address Register (XDMAC_CDAx) for channel x.
5. Program field UBLEN in the XDMAC Channel x Microblock Control Register (XDMAC_CUBCx) with the number of data.
6. Program the XDMAC Channel x Configuration Register (XDMAC_CCx):
   a. Clear XDMAC_CCx.TYPE for a memory to memory transfer, otherwise set this bit.
   b. Program XDMAC_CCx.MBSIZE to the memory burst size used.
   c. Program XDMAC_CCx.SAM/DAM to the memory addressing scheme.
   d. Program XDMAC_CCx.SYNC to select the peripheral transfer direction.
   e. Program XDMAC_CCx.CSIZE to configure the channel chunk size (only relevant for peripheral synchronized transfer).
   f. Program XDMAC_CCx.DWIDTH to configure the transfer data width.
   g. Program XDMAC_CCx.SIF, XDMAC_CCx.DIF to configure the master interface used to read data and write data respectively.
   h. Program XDMAC_CCx.PERID to select the active hardware request line (only relevant for a peripheral synchronized transfer).
   i. Set XDMAC_CCx.SWREQ to use software request (only relevant for a peripheral synchronized transfer).
7. Clear the following five registers:
   XDMAC Channel x Next Descriptor Control Register (XDMAC_CNDCx)
   XDMAC Channel x Block Control Register (XDMAC_CBCx)
   XDMAC Channel x Data Stride Memory Set Pattern Register (XDMAC_CDS_MSPx)
   XDMAC Channel x Source Microblock Stride Register (XDMAC_CSUSx)
   XDMAC Channel x Destination Microblock Stride Register (XDMAC_CDUSx)
   This respectively indicates that the linked list is disabled, there is only one block and striding is disabled.
8. Enable the Microblock interrupt by writing a 1 to bit BIE in the XDMAC Channel x Interrupt Enable Register (XDMAC_CIEx), enable the Channel x Interrupt Enable bit by writing a 1 to bit IEx in the XDMAC Global Interrupt Enable Register (XDMAC_GIE).
9. Enable channel x by writing a 1 to bit ENx in the XDMAC Global Channel Enable Register (XDMAC_GE). XDMAC_GS.STx (XDMAC Channel x Status bit) is set by hardware.

Downloaded from Arrow.com.

10. Once completed, the DMA channel sets XDMAC_CISx.BIS (End of Block Interrupt Status bit) and generates an interrupt. XDMAC_GS.STx is cleared by hardware. The software can either wait for an interrupt or poll the channel status bit.

35.5.4.2 Single Block Transfer With Multiple Microblock

1. Read the XDMAC_GS register to choose a free channel.
2. Clear the pending Interrupt Status bit by reading the chosen XDMAC_CISx register.
3. Write the XDMAC_CSAx register for channel x.
4. Write the XDMAC_CDAx register for channel x.
5. Program XDMAC_CUBCx.UBLEN with the number of data.
6. Program XDMAC_CCx register (see single block transfer configuration).
7. Program XDMAC_CBCx.BLEN with the number of microblocks of data.
8. Clear the following four registers:
   XDMAC_CNDCx
   XDMAC_CDS_MSPx
   XDMAC_CSUSx
   XDMAC_CDUSx
   This respectively indicates that the linked list is disabled and striding is disabled.
9. Enable the Block interrupt by writing a 1 to XDMAC_CIEx.BIE, enable the Channel x Interrupt Enable bit by writing a 1 to XDMAC_GIEx.IEx.
10. Enable channel x by writing a 1 to the XDMAC_GE.ENx. XDMAC_GS.STx is set by hardware.
11. Once completed, the DMA channel sets XDMAC_CISx.BIS (End of Block Interrupt Status bit) and generates an interrupt. XDMAC_GS.STx is cleared by hardware. The software can either wait for an interrupt or poll the channel status bit.

35.5.4.3 Master Transfer

1. Read the XDMAC_GS register to choose a free channel.
2. Clear the pending Interrupt Status bit by reading the chosen XDMAC_CISx register.
3. Build a linked list of transfer descriptors in memory. The descriptor view is programmable on a per descriptor basis. The linked list items structure must be word aligned. MBR_UBC.NDE must be configured to 0 in the last descriptor to terminate the list.
4. Program field NDA in the XDMAC Channel x Next Descriptor Address Register (XDMAC_CNDAx) with the first descriptor address and bit XDMAC_CNDAx.NDAIF with the master interface identifier.
5. Program the XDMAC_CNDCx register:
   a. Set XDMAC_CNDCx.NDE to enable the descriptor fetch.
   b. Set XDMAC_CNDCx.NDSUP to update the source address at the descriptor fetch time, otherwise clear this bit.
   c. Set XDMAC_CNDCx.NDDUP to update the destination address at the descriptor fetch time, otherwise clear this bit.
   d. Program XDMAC_CNDCx.NDVIEW to define the length of the first descriptor.
6. Enable the End of Linked List interrupt by writing a 1 to XDMAC_CIEx.LIE.
7. Enable channel x by writing a 1 to XDMAC_GE.ENx. XDMAC_GS.STx is set by hardware.
8. Once completed, the DMA channel sets XDMAC_CISx.BIS (End of Block Interrupt Status bit) and generates an interrupt. XDMAC_GS.STx is cleared by hardware. The software can either wait for an interrupt or poll the channel status bit.

Downloaded from Arrow.com.

35.5.4.4 Disabling A Channel Before Transfer Completion

Under normal operation, the software enables a channel by writing a 1 to XDMAC_GE.ENx (Global Channel x Enable Register bit), then the hardware disables a channel on transfer completion by clearing bit XDMAC_GS.STx. To disable a channel, write a 1 to bit XDMAC_GD.DIx and poll the XDMAC_GS register.

Downloaded from Arrow.com.

35.6 Linked List Descriptor Operation

35.6.1 Linked List Descriptor View

35.6.1.1 Channel Next Descriptor View 0–3 Structures

Table 35-2. Channel Next Descriptor View 0–3 Structures

Channel Next Descriptor	Offset	Structure member	Name
View 0 Structure	DSCR_ADDR+0x00	Next Descriptor Address Member	MBR_NDA
	DSCR_ADDR+0x04	Microblock Control Member	MBR_UBC
	DSCR_ADDR+0x08	Transfer Address Member	MBR_TA
View 1 Structure	DSCR_ADDR+0x00	Next Descriptor Address Member	MBR_NDA
	DSCR_ADDR+0x04	Microblock Control Member	MBR_UBC
	DSCR_ADDR+0x08	Source Address Member	MBR_SA
	DSCR_ADDR+0x0C	Destination Address Member	MBR_DA
View 2 Structure	DSCR_ADDR+0x00	Next Descriptor Address Member	MBR_NDA
	DSCR_ADDR+0x04	Microblock Control Member	MBR_UBC
	DSCR_ADDR+0x08	Source Address Member	MBR_SA
	DSCR_ADDR+0x0C	Destination Address Member	MBR_DA
	DSCR_ADDR+0x10	Configuration Register	MBR_CFG
View 3 Structure	DSCR_ADDR+0x00	Next Descriptor Address Member	MBR_NDA
	DSCR_ADDR+0x04	Microblock Control Member	MBR_UBC
	DSCR_ADDR+0x08	Source Address Member	MBR_SA
	DSCR_ADDR+0x0C	Destination Address Member	MBR_DA
	DSCR_ADDR+0x10	Configuration Member	MBR_CFG
	DSCR_ADDR+0x14	Block Control Member	MBR_BC
	DSCR_ADDR+0x18	Data Stride Member	MBR_DS
	DSCR_ADDR+0x1C	Source Microblock Stride Member	MBR_SUS
	DSCR_ADDR+0x20	Destination Microblock Stride Member	MBR_DUS

Downloaded from Arrow.com.

35.6.2 Descriptor Structure Members Description

35.6.2.1 Descriptor Structure Microblock Control Member

Name: MBR_UBC

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	NVIEW		NDEN	NSEN	NDE
23	22	21	20	19	18	17	16
UBLEN							
15	14	13	12	11	10	9	8
UBLEN							
7	6	5	4	3	2	1	0
UBLEN							

- **UBLEN: Microblock Length**

This field indicates the number of data in the microblock. The microblock contains UBLEN data.

- **NDE: Next Descriptor Enable**

0: Descriptor fetch is disabled.

1: Descriptor fetch is enabled.

- **NSEN: Next Descriptor Source Update**

0: Source parameters remain unchanged.

1: Source parameters are updated when the descriptor is retrieved.

- **NDEN: Next Descriptor Destination Update**

0: Destination parameters remain unchanged.

1: Destination parameters are updated when the descriptor is retrieved.

- **NVIEW: Next Descriptor View**

Value	Name	Description
0	NDV0	Next Descriptor View 0
1	NDV1	Next Descriptor View 1
2	NDV2	Next Descriptor View 2
3	NDV3	Next Descriptor View 3

Downloaded from Arrow.com.

35.7 XDMAC Maintenance Software Operations

35.7.1 Disabling a Channel

A disable channel request occurs when a write operation is performed in the XDMAC_GD register. If the channel is source peripheral synchronized (bit XDMAC_CCx.TYPE is set and bit XDMAC_CCx.DSYNC is cleared), then pending bytes (bytes located in the FIFO) are written to memory and bit XDMAC_CISx.DIS is set. If the channel is not source peripheral synchronized, the current channel transaction (read or write) is terminated and XDMAC_CISx.DIS is set. XDMAC_GS.STx is cleared by hardware when the current transfer is completed. The channel is no longer active and can be reused.

35.7.2 Suspending a Channel

A read request suspend command is issued by writing to the XDMAC_GRS register. A write request suspend command is issued by writing to the XDMAC_GWS register. A read write suspend channel is issued by writing to the XDMAC_GRWS register. These commands have an immediate effect on the scheduling of both read and write transactions. If a transaction is already in progress, it is terminated normally. The channel is not disabled. The FIFO content is preserved. The scheduling mechanism can resume normally, clearing the bit in the same registers. Pending bytes located in the FIFO are not written out to memory. The write suspend command does not affect read request operations, i.e., read operations can still occur until the FIFO is full.

35.7.3 Flushing a Channel

A FIFO flush command is issued writing to the XDMAC_SWF register. The content of the FIFO is written to memory. XDMAC_CISx.FIS (End of Flush Interrupt Status bit) is set when the last byte is successfully transferred to memory. The channel is not disabled. The flush operation is not blocking, meaning that read operation can be scheduled during the flush write operation. The flush operation is only relevant for peripheral to memory transfer where pending peripheral bytes are buffered into the channel FIFO.

35.7.4 Maintenance Operation Priority

35.7.4.1 Disable Operation Priority

- When a disable request occurs on a suspended channel, the XDMAC_GWS.WSx (Channel x Write Suspend bit) is cleared. If the transfer is source peripheral synchronized, the pending bytes are drained to memory. The bit XDMAC_CISx.DIS is set.
- When a disable request follows a flush request, if the flush last transaction is not yet scheduled, the flush request is discarded and the disable procedure is applied. The bit XDMAC_CISx.FIS is not set. Bit XDMAC_CISx.DIS will be set when the disable request is completed. If the flush request transaction is already scheduled, the XDMAC_CISx.FIS will be set. XDMAC_CISx.DIS will also be set when the disable request is completed.

35.7.4.2 Flush Operation Priority

- When a flush request occurs on a suspended channel, if there are pending bytes in the FIFO, they are written out to memory, XDMAC_CISx.FIS is set. If the FIFO is empty, XDMAC_CISx.FIS is also set.
- If the flush operation is performed after a disable request, the flush command is ignored. XDMAC_CISx.FIS is not set.

35.7.4.3 Suspend Operation Priority

If the suspend operation is performed after a disable request, the write suspend operation is ignored.

Downloaded from Arrow.com.

35.8 XDMAC Software Requirements

- Write operations to channel registers are not be performed in an active channel after the channel is enabled. If any channel parameters must be reprogrammed, this can only be done after disabling the XDMAC channel.
- XDMAC_CSAx and XDMAC_CDAx channel registers are to be programmed with a byte, half-word or word aligned address depending on the Channel x Data Width field (DWIDTH) of the XDMAC Channel x Configuration Register.

Downloaded from Arrow.com.

35.9 Extensible DMA Controller (XDMAC) User Interface

Table 35-3. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Global Type Register	XDMAC_GTYPE	Read-only	0x00000000
0x04	Global Configuration Register	XDMAC_GCFG	Read-only	0x00000000
0x08	Global Weighted Arbiter Configuration Register	XDMAC_GWAC	Read-only	0x00000000
0x0C	Global Interrupt Enable Register	XDMAC_GIE	Write-only	–
0x10	Global Interrupt Disable Register	XDMAC_GID	Write-only	–
0x14	Global Interrupt Mask Register	XDMAC_GIM	Read-only	0x00000000
0x18	Global Interrupt Status Register	XDMAC_GIS	Read-only	0x00000000
0x1C	Global Channel Enable Register	XDMAC_GE	Write-only	–
0x20	Global Channel Disable Register	XDMAC_GD	Write-only	–
0x24	Global Channel Status Register	XDMAC_GS	Read-only	0x00000000
0x28	Global Channel Read Suspend Register	XDMAC_GRS	Read/Write	0x00000000
0x2C	Global Channel Write Suspend Register	XDMAC_GWS	Read/Write	0x00000000
0x30	Global Channel Read Write Suspend Register	XDMAC_GRWS	Write-only	–
0x34	Global Channel Read Write Resume Register	XDMAC_GRWR	Write-only	–
0x38	Global Channel Software Request Register	XDMAC_GSWR	Write-only	–
0x3C	Global Channel Software Request Status Register	XDMAC_GSWS	Read-only	0x00000000
0x40	Global Channel Software Flush Request Register	XDMAC_GSWF	Write-only	–
0x44–0x4C	Reserved	–	–	–
0x50+chid*0x40	Channel Interrupt Enable Register	XDMAC_CIE	Write-only	–
0x54+chid*0x40	Channel Interrupt Disable Register	XDMAC_CID	Write-only	–
0x58+chid*0x40	Channel Interrupt Mask Register	XDMAC_CIM	Read-only	–
0x5C+chid*0x40	Channel Interrupt Status Register	XDMAC_CIS	Read-only	0x00000000
0x60+chid*0x40	Channel Source Address Register	XDMAC_CSA	Read/Write	0x00000000
0x64+chid*0x40	Channel Destination Address Register	XDMAC_CDA	Read/Write	0x00000000
0x68+chid*0x40	Channel Next Descriptor Address Register	XDMAC_CNDA	Read/Write	0x00000000
0x6C+chid*0x40	Channel Next Descriptor Control Register	XDMAC_CNDC	Read/Write	0x00000000
0x70+chid*0x40	Channel Microblock Control Register	XDMAC_CUBC	Read/Write	0x00000000
0x74+chid*0x40	Channel Block Control Register	XDMAC_CBC	Read/Write	0x00000000
0x78+chid*0x40	Channel Configuration Register	XDMAC_CC	Read/Write	0x00000000
0x7C+chid*0x40	Channel Data Stride Memory Set Pattern	XDMAC_CDS_MSP	Read/Write	0x00000000
0x80+chid*0x40	Channel Source Microblock Stride	XDMAC_CSUS	Read/Write	0x00000000
0x84+chid*0x40	Channel Destination Microblock Stride	XDMAC_CDUS	Read/Write	0x00000000
0x88+chid*0x40	Reserved	–	–	–
0x8C+chid*0x40	Reserved	–	–	–
0xFEC–0xFFC	Reserved	–	–	–

Downloaded from Arrow.com.

35.9.1 XDMAC Global Type Register

Name: XDMAC_GTYPE

Address: 0x40078000

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	NB_REQ						

15	14	13	12	11	10	9	8
FIFO_SZ							

7	6	5	4	3	2	1	0
FIFO_SZ			NB_CH				

- **NB_CH: Number of Channels Minus One**
- **FIFO_SZ: Number of Bytes**
- **NB_REQ: Number of Peripheral Requests Minus One**

Downloaded from Arrow.com.

35.9.2 XDMAC Global Configuration Register

Name: XDMAC_GCFG

Address: 0x40078004

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	BXKBEN
7	6	5	4	3	2	1	0
–	–	–	–	CGDISIF	CGDISFIFO	CGDISPIPE	CGDISREG

- **CGDISREG: Configuration Registers Clock Gating Disable**

0: The automatic clock gating is enabled for the configuration registers.

1: The automatic clock gating is disabled for the configuration registers.

- **CGDISPIPE: Pipeline Clock Gating Disable**

0: The automatic clock gating is enabled for the main pipeline.

1: The automatic clock gating is disabled for the main pipeline.

- **CGDISFIFO: FIFO Clock Gating Disable**

0: The automatic clock gating is enabled for the main FIFO.

1: The automatic clock gating is disabled for the main FIFO.

- **CGDISIF: Bus Interface Clock Gating Disable**

0: The automatic clock gating is enabled for the system bus interface.

1: The automatic clock gating is disabled for the system bus interface.

- **BXKBEN: Boundary X Kilobyte Enable**

0: The 1 Kbyte boundary is used.

1: The controller does not meet the AHB specification.

Downloaded from Arrow.com.

35.9.3 XDMAC Global Weighted Arbiter Configuration Register

Name: XDMAC_GWAC

Address: 0x40078008

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
PW3				PW2			

7	6	5	4	3	2	1	0
PW1				PW0			

- **PW0: Pool Weight 0**

This field indicates the weight of the pool 0, in the arbitration scheme of the XDMA scheduler.

- **PW1: Pool Weight 1**

This field indicates the weight of the pool 1, in the arbitration scheme of the XDMA scheduler.

- **PW2: Pool Weight 2**

This field indicates the weight of the pool 2, in the arbitration scheme of the XDMA scheduler.

- **PW3: Pool Weight 3**

This field indicates the weight of the pool 3, in the arbitration scheme of the XDMA scheduler.

Downloaded from Arrow.com.

35.9.4 XDMAC Global Interrupt Enable Register

Name: XDMAC_GIE

Address: 0x4007800C

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
IE23	IE22	IE21	IE20	IE19	IE18	IE17	IE16
15	14	13	12	11	10	9	8
IE15	IE14	IE13	IE12	IE11	IE10	IE9	IE8
7	6	5	4	3	2	1	0
IE7	IE6	IE5	IE4	IE3	IE2	IE1	IE0

- **IEx: XDMAC Channel x Interrupt Enable Bit**

0: This bit has no effect. The channel x Interrupt Mask bit is not modified.

1: The corresponding mask bit is set. The XDMAC Channel x Interrupt Status Register can generate an interrupt.

Downloaded from Arrow.com.

35.9.5 XDMAC Global Interrupt Disable Register

Name: XDMAC_GID

Address: 0x40078010

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
15	14	13	12	11	10	9	8
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
7	6	5	4	3	2	1	0
ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0

- **IDx: XDMAC Channel x Interrupt Disable Bit**

0: This bit has no effect. The channel x Interrupt Mask bit is not modified.

1: The corresponding mask bit is reset. The Channel x interrupt status register interrupt is masked.

Downloaded from Arrow.com.

35.9.6 XDMAC Global Interrupt Mask Register

Name: XDMAC_GIM

Address: 0x40078014

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
15	14	13	12	11	10	9	8
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8
7	6	5	4	3	2	1	0
IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0

- **IMx: XDMAC Channel x Interrupt Mask Bit**

0: This bit indicates that the channel x interrupt source is masked. The interrupt line is not raised.

1: This bit indicates that the channel x interrupt source is unmasked.

Downloaded from Arrow.com.

35.9.7 XDMAC Global Interrupt Status Register

Name: XDMAC_GIS

Address: 0x40078018

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
IS23	IS22	IS21	IS20	IS19	IS18	IS17	IS16
15	14	13	12	11	10	9	8
IS15	IS14	IS13	IS12	IS11	IS10	IS9	IS8
7	6	5	4	3	2	1	0
IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0

- **ISx: XDMAC Channel x Interrupt Status Bit**

0: This bit indicates that either the interrupt source is masked at the channel level or no interrupt is pending for channel x.

1: This bit indicates that an interrupt is pending for the channel x.

Downloaded from Arrow.com.

35.9.8 XDMAC Global Channel Enable Register

Name: XDMAC_GE

Address: 0x4007801C

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
EN23	EN22	EN21	EN20	EN19	EN18	EN17	EN16
15	14	13	12	11	10	9	8
EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8
7	6	5	4	3	2	1	0
EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0

- **ENx: XDMAC Channel x Enable Bit**

0: This bit has no effect.

1: Enables channel x. This operation is permitted if the channel x status bit was read as 0.

Downloaded from Arrow.com.

35.9.9 XDMAC Global Channel Disable Register

Name: XDMAC_GD

Address: 0x40078020

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
DI23	DI22	DI21	DI20	DI19	DI18	DI17	DI16

15	14	13	12	11	10	9	8
DI15	DI14	DI13	DI12	DI11	DI10	DI9	DI8

7	6	5	4	3	2	1	0
DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0

- **DIx: XDMAC Channel x Disable Bit**

0: This bit has no effect.

1: Disables channel x.

Downloaded from Arrow.com.

35.9.10 XDMAC Global Channel Status Register

Name: XDMAC_GS

Address: 0x40078024

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
ST23	ST22	ST21	ST20	ST19	ST18	ST17	ST16
15	14	13	12	11	10	9	8
ST15	ST14	ST13	ST12	ST11	ST10	ST9	ST8
7	6	5	4	3	2	1	0
ST7	ST6	ST5	ST4	ST3	ST2	ST1	ST0

- **STx: XDMAC Channel x Status Bit**

0: This bit indicates that the channel x is disabled.

1: This bit indicates that the channel x is enabled. If a channel disable request is issued, this bit remains asserted until pending transaction is completed.

Downloaded from Arrow.com.

35.9.11 XDMAC Global Channel Read Suspend Register

Name: XDMAC_GRS

Address: 0x40078028

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
RS23	RS22	RS21	RS20	RS19	RS18	RS17	RS16
15	14	13	12	11	10	9	8
RS15	RS14	RS13	RS12	RS11	RS10	RS9	RS8
7	6	5	4	3	2	1	0
RS7	RS6	RS5	RS4	RS3	RS2	RS1	RS0

- **RSx: XDMAC Channel x Read Suspend Bit**

0: The read channel is not suspended.

1: The source requests for channel x are no longer serviced by the system scheduler.

Downloaded from Arrow.com.

35.9.12 XDMAC Global Channel Write Suspend Register

Name: XDMAC_GWS

Address: 0x4007802C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
WS23	WS22	WS21	WS20	WS19	WS18	WS17	WS16
15	14	13	12	11	10	9	8
WS15	WS14	WS13	WS12	WS11	WS10	WS9	WS8
7	6	5	4	3	2	1	0
WS7	WS6	WS5	WS4	WS3	WS2	WS1	WS0

- **WSx: XDMAC Channel x Write Suspend Bit**

0: The write channel is not suspended.

1: Destination requests are no longer routed to the scheduler.

Downloaded from Arrow.com.

35.9.13 XDMAC Global Channel Read Write Suspend Register

Name: XDMAC_GRWS

Address: 0x40078030

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
RWS23	RWS22	RWS21	RWS20	RWS19	RWS18	RWS17	RWS16
15	14	13	12	11	10	9	8
RWS15	RWS14	RWS13	RWS12	RWS11	RWS10	RWS9	RWS8
7	6	5	4	3	2	1	0
RWS7	RWS6	RWS5	RWS4	RWS3	RWS2	RWS1	RWS0

- **RWSx: XDMAC Channel x Read Write Suspend Bit**

0: No effect.

1: Read and Write requests are suspended.

Downloaded from Arrow.com.

35.9.14 XDMAC Global Channel Read Write Resume Register

Name: XDMAC_GRWR

Address: 0x40078034

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
RWR23	RWR22	RWR21	RWR20	RWR19	RWR18	RWR17	RWR16

15	14	13	12	11	10	9	8
RWR15	RWR14	RWR13	RWR12	RWR11	RWR10	RWR9	RWR8

7	6	5	4	3	2	1	0
RWR7	RWR6	RWR5	RWR4	RWR3	RWR2	RWR1	RWR0

- **RWRx: XDMAC Channel x Read Write Resume Bit**

0: No effect.

1: Read and Write requests are serviced.

Downloaded from Arrow.com.

35.9.15 XDMAC Global Channel Software Request Register

Name: XDMAC_GSWR

Address: 0x40078038

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
SWREQ23	SWREQ22	SWREQ21	SWREQ20	SWREQ19	SWREQ18	SWREQ17	SWREQ16

15	14	13	12	11	10	9	8
SWREQ15	SWREQ14	SWREQ13	SWREQ12	SWREQ11	SWREQ10	SWREQ9	SWREQ8

7	6	5	4	3	2	1	0
SWREQ7	SWREQ6	SWREQ5	SWREQ4	SWREQ3	SWREQ2	SWREQ1	SWREQ0

- **SWREQx: XDMAC Channel x Software Request Bit**

0: No effect.

1: Requests a DMA transfer for channel x.

Downloaded from Arrow.com.

35.9.16 XDMAC Global Channel Software Request Status Register

Name: XDMAC_GSWS

Address: 0x4007803C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
SWRS23	SWRS22	SWRS21	SWRS20	SWRS19	SWRS18	SWRS17	SWRS16

15	14	13	12	11	10	9	8
SWRS15	SWRS14	SWRS13	SWRS12	SWRS11	SWRS10	SWRS9	SWRS8

7	6	5	4	3	2	1	0
SWRS7	SWRS6	SWRS5	SWRS4	SWRS3	SWRS2	SWRS1	SWRS0

- **SWRSx: XDMAC Channel x Software Request Status Bit**

0: Channel x Source request is serviced.

1: Channel x Source request is pending.

Downloaded from Arrow.com.

35.9.17 XDMAC Global Channel Software Flush Request Register

Name: XDMAC_GSWF

Address: 0x40078040

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
SWF23	SWF22	SWF21	SWF20	SWF19	SWF18	SWF17	SWF16

15	14	13	12	11	10	9	8
SWF15	SWF14	SWF13	SWF12	SWF11	SWF10	SWF9	SWF8

7	6	5	4	3	2	1	0
SWF7	SWF6	SWF5	SWF4	SWF3	SWF2	SWF1	SWF0

- **SWFx: XDMAC Channel x Software Flush Request Bit**

0: No effect.

1: Requests a DMA transfer flush for channel x. This bit is only relevant when the transfer is source peripheral synchronized.

Downloaded from Arrow.com.

35.9.18 XDMAC Channel x [x = 0..23] Interrupt Enable Register

Name: XDMAC_CIEx [x = 0..23]

Address: 0x40078050 [0], 0x40078090 [1], 0x400780D0 [2], 0x40078110 [3], 0x40078150 [4], 0x40078190 [5], 0x400781D0 [6], 0x40078210 [7], 0x40078250 [8], 0x40078290 [9], 0x400782D0 [10], 0x40078310 [11], 0x40078350 [12], 0x40078390 [13], 0x400783D0 [14], 0x40078410 [15], 0x40078450 [16], 0x40078490 [17], 0x400784D0 [18], 0x40078510 [19], 0x40078550 [20], 0x40078590 [21], 0x400785D0 [22], 0x40078610 [23]

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	ROIE	WBIE	RBIE	FIE	DIE	LIE	BIE

- **BIE: End of Block Interrupt Enable Bit**

0: No effect.

1: Enables end of block interrupt.

- **LIE: End of Linked List Interrupt Enable Bit**

0: No effect.

1: Enables end of linked list interrupt.

- **DIE: End of Disable Interrupt Enable Bit**

0: No effect.

1: Enables end of disable interrupt.

- **FIE: End of Flush Interrupt Enable Bit**

0: No effect.

1: Enables end of flush interrupt.

- **RBIE: Read Bus Error Interrupt Enable Bit**

0: No effect.

1: Enables read bus error interrupt.

- **WBIE: Write Bus Error Interrupt Enable Bit**

0: No effect.

1: Enables write bus error interrupt.

- **ROIE: Request Overflow Error Interrupt Enable Bit**

0: No effect.

1: Enables Request Overflow Error Interrupt.

Downloaded from Arrow.com.

35.9.19 XDMAC Channel x [x = 0..23] Interrupt Disable Register

Name: XDMAC_CIDx [x = 0..23]

Address: 0x40078054 [0], 0x40078094 [1], 0x400780D4 [2], 0x40078114 [3], 0x40078154 [4], 0x40078194 [5], 0x400781D4 [6], 0x40078214 [7], 0x40078254 [8], 0x40078294 [9], 0x400782D4 [10], 0x40078314 [11], 0x40078354 [12], 0x40078394 [13], 0x400783D4 [14], 0x40078414 [15], 0x40078454 [16], 0x40078494 [17], 0x400784D4 [18], 0x40078514 [19], 0x40078554 [20], 0x40078594 [21], 0x400785D4 [22], 0x40078614 [23]

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	ROID	WBEID	RBEID	FID	DID	LID	BID

- **BID: End of Block Interrupt Disable Bit**

0: No effect.

1: Disables end of block interrupt.

- **LID: End of Linked List Interrupt Disable Bit**

0: No effect.

1: Disables end of linked list interrupt.

- **DID: End of Disable Interrupt Disable Bit**

0: No effect.

1: Disables end of disable interrupt.

- **FID: End of Flush Interrupt Disable Bit**

0: No effect.

1: Disables end of flush interrupt.

- **RBEID: Read Bus Error Interrupt Disable Bit**

0: No effect.

1: Disables bus error interrupt.

- **WBEID: Write Bus Error Interrupt Disable Bit**

0: No effect.

1: Disables bus error interrupt.

- **ROID: Request Overflow Error Interrupt Disable Bit**

0: No effect.

1: Disables Request Overflow Error Interrupt.

Downloaded from Arrow.com.

35.9.20 XDMAC Channel x [x = 0..23] Interrupt Mask Register

Name: XDMAC_CIMx [x = 0..23]

Address: 0x40078058 [0], 0x40078098 [1], 0x400780D8 [2], 0x40078118 [3], 0x40078158 [4], 0x40078198 [5], 0x400781D8 [6], 0x40078218 [7], 0x40078258 [8], 0x40078298 [9], 0x400782D8 [10], 0x40078318 [11], 0x40078358 [12], 0x40078398 [13], 0x400783D8 [14], 0x40078418 [15], 0x40078458 [16], 0x40078498 [17], 0x400784D8 [18], 0x40078518 [19], 0x40078558 [20], 0x40078598 [21], 0x400785D8 [22], 0x40078618 [23]

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	ROIM	WBEIM	RBEIM	FIM	DIM	LIM	BIM

- **BIM: End of Block Interrupt Mask Bit**

0: Block interrupt is masked.

1: Block interrupt is activated.

- **LIM: End of Linked List Interrupt Mask Bit**

0: End of linked list interrupt is masked.

1: End of linked list interrupt is activated.

- **DIM: End of Disable Interrupt Mask Bit**

0: End of disable interrupt is masked.

1: End of disable interrupt is activated.

- **FIM: End of Flush Interrupt Mask Bit**

0: End of flush interrupt is masked.

1: End of flush interrupt is activated.

- **RBEIM: Read Bus Error Interrupt Mask Bit**

0: Bus error interrupt is masked.

1: Bus error interrupt is activated.

- **WBEIM: Write Bus Error Interrupt Mask Bit**

0: Bus error interrupt is masked.

1: Bus error interrupt is activated.

- **ROIM: Request Overflow Error Interrupt Mask Bit**

0: Request Overflow interrupt is masked.

1: Request Overflow interrupt is activated.

Downloaded from Arrow.com.

35.9.21 XDMAC Channel x [x = 0..23] Interrupt Status Register

Name: XDMAC_CISx [x = 0..23]

Address: 0x4007805C [0], 0x4007809C [1], 0x400780DC [2], 0x4007811C [3], 0x4007815C [4], 0x4007819C [5], 0x400781DC [6], 0x4007821C [7], 0x4007825C [8], 0x4007829C [9], 0x400782DC [10], 0x4007831C [11], 0x4007835C [12], 0x4007839C [13], 0x400783DC [14], 0x4007841C [15], 0x4007845C [16], 0x4007849C [17], 0x400784DC [18], 0x4007851C [19], 0x4007855C [20], 0x4007859C [21], 0x400785DC [22], 0x4007861C [23]

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	ROIS	WBEIS	RBEIS	FIS	DIS	LIS	BIS

- **BIS: End of Block Interrupt Status Bit**

0: End of block interrupt has not occurred.

1: End of block interrupt has occurred since the last read of the status register.

- **LIS: End of Linked List Interrupt Status Bit**

0: End of linked list condition has not occurred.

1: End of linked list condition has occurred since the last read of the status register.

- **DIS: End of Disable Interrupt Status Bit**

0: End of disable condition has not occurred.

1: End of disable condition has occurred since the last read of the status register.

- **FIS: End of Flush Interrupt Status Bit**

0: End of flush condition has not occurred.

1: End of flush condition has occurred since the last read of the status register.

- **RBEIS: Read Bus Error Interrupt Status Bit**

0: Read bus error condition has not occurred.

1: At least one bus error has been detected in a read access since the last read of the status register.

- **WBEIS: Write Bus Error Interrupt Status Bit**

0: Write bus error condition has not occurred.

1: At least one bus error has been detected in a write access since the last read of the status register.

- **ROIS: Request Overflow Error Interrupt Status Bit**

0: Overflow condition has not occurred.

1: Overflow condition has occurred at least once. (This information is only relevant for peripheral synchronized transfers.)

Downloaded from Arrow.com.

35.9.22 XDMAC Channel x [x = 0..23] Source Address Register

Name: XDMAC_CSAx [x = 0..23]

Address: 0x40078060 [0], 0x400780A0 [1], 0x400780E0 [2], 0x40078120 [3], 0x40078160 [4], 0x400781A0 [5], 0x400781E0 [6], 0x40078220 [7], 0x40078260 [8], 0x400782A0 [9], 0x400782E0 [10], 0x40078320 [11], 0x40078360 [12], 0x400783A0 [13], 0x400783E0 [14], 0x40078420 [15], 0x40078460 [16], 0x400784A0 [17], 0x400784E0 [18], 0x40078520 [19], 0x40078560 [20], 0x400785A0 [21], 0x400785E0 [22], 0x40078620 [23]

Access: Read/Write

31	30	29	28	27	26	25	24
SA							

23	22	21	20	19	18	17	16
SA							

15	14	13	12	11	10	9	8
SA							

7	6	5	4	3	2	1	0
SA							

- **SA: Channel x Source Address**

Program this register with the source address of the DMA transfer.

A configuration error is generated when this address is not aligned with the transfer data size.

Downloaded from Arrow.com.

35.9.23 XDMAC Channel x [x = 0..23] Destination Address Register

Name: XDMAC_CDAx [x = 0..23]

Address: 0x40078064 [0], 0x400780A4 [1], 0x400780E4 [2], 0x40078124 [3], 0x40078164 [4], 0x400781A4 [5], 0x400781E4 [6], 0x40078224 [7], 0x40078264 [8], 0x400782A4 [9], 0x400782E4 [10], 0x40078324 [11], 0x40078364 [12], 0x400783A4 [13], 0x400783E4 [14], 0x40078424 [15], 0x40078464 [16], 0x400784A4 [17], 0x400784E4 [18], 0x40078524 [19], 0x40078564 [20], 0x400785A4 [21], 0x400785E4 [22], 0x40078624 [23]

Access: Read/Write

31	30	29	28	27	26	25	24
DA							

23	22	21	20	19	18	17	16
DA							

15	14	13	12	11	10	9	8
DA							

7	6	5	4	3	2	1	0
DA							

- **DA: Channel x Destination Address**

Program this register with the destination address of the DMA transfer.

A configuration error is generated when this address is not aligned with the transfer data size.

Downloaded from Arrow.com.

35.9.24 XDMAC Channel x [x = 0..23] Next Descriptor Address Register

Name: XDMAC_CNDAx [x = 0..23]

Address: 0x40078068 [0], 0x400780A8 [1], 0x400780E8 [2], 0x40078128 [3], 0x40078168 [4], 0x400781A8 [5], 0x400781E8 [6], 0x40078228 [7], 0x40078268 [8], 0x400782A8 [9], 0x400782E8 [10], 0x40078328 [11], 0x40078368 [12], 0x400783A8 [13], 0x400783E8 [14], 0x40078428 [15], 0x40078468 [16], 0x400784A8 [17], 0x400784E8 [18], 0x40078528 [19], 0x40078568 [20], 0x400785A8 [21], 0x400785E8 [22], 0x40078628 [23]

Access: Read/Write

31	30	29	28	27	26	25	24
NDA							

23	22	21	20	19	18	17	16
NDA							

15	14	13	12	11	10	9	8
NDA							

7	6	5	4	3	2	1	0
NDA						–	NDAIF

- **NDAIF: Channel x Next Descriptor Interface**

0: The channel descriptor is retrieved through the system interface 0.

1: The channel descriptor is retrieved through the system interface 1.

- **NDA: Channel x Next Descriptor Address**

The 30-bit width of the NDA field represents the next descriptor address range 31:2. The descriptor is word-aligned and the two least significant register bits 1:0 are ignored.

Downloaded from Arrow.com.

35.9.25 XDMAC Channel x [x = 0..23] Next Descriptor Control Register

Name: XDMAC_CNDCx [x = 0..23]

Address: 0x4007806C [0], 0x400780AC [1], 0x400780EC [2], 0x4007812C [3], 0x4007816C [4], 0x400781AC [5], 0x400781EC [6], 0x4007822C [7], 0x4007826C [8], 0x400782AC [9], 0x400782EC [10], 0x4007832C [11], 0x4007836C [12], 0x400783AC [13], 0x400783EC [14], 0x4007842C [15], 0x4007846C [16], 0x400784AC [17], 0x400784EC [18], 0x4007852C [19], 0x4007856C [20], 0x400785AC [21], 0x400785EC [22], 0x4007862C [23]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	NDVIEW		NDDUP	NDSUP	NDE

- **NDE: Channel x Next Descriptor Enable**

0 (DSCR_FETCH_DIS): Descriptor fetch is disabled.

1 (DSCR_FETCH_EN): Descriptor fetch is enabled.

- **NDSUP: Channel x Next Descriptor Source Update**

0 (SRC_PARAMS_UNCHANGED): Source parameters remain unchanged.

1 (SRC_PARAMS_UPDATED): Source parameters are updated when the descriptor is retrieved.

- **NDDUP: Channel x Next Descriptor Destination Update**

0 (DST_PARAMS_UNCHANGED): Destination parameters remain unchanged.

1 (DST_PARAMS_UPDATED): Destination parameters are updated when the descriptor is retrieved.

- **NDVIEW: Channel x Next Descriptor View**

Value	Name	Description
0	NDV0	Next Descriptor View 0
1	NDV1	Next Descriptor View 1
2	NDV2	Next Descriptor View 2
3	NDV3	Next Descriptor View 3

Downloaded from Arrow.com.

35.9.26 XDMAC Channel x [x = 0..23] Microblock Control Register

Name: XDMAC_CUBCx [x = 0..23]

Address: 0x40078070 [0], 0x400780B0 [1], 0x400780F0 [2], 0x40078130 [3], 0x40078170 [4], 0x400781B0 [5], 0x400781F0 [6], 0x40078230 [7], 0x40078270 [8], 0x400782B0 [9], 0x400782F0 [10], 0x40078330 [11], 0x40078370 [12], 0x400783B0 [13], 0x400783F0 [14], 0x40078430 [15], 0x40078470 [16], 0x400784B0 [17], 0x400784F0 [18], 0x40078530 [19], 0x40078570 [20], 0x400785B0 [21], 0x400785F0 [22], 0x40078630 [23]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
UBLEN							

15	14	13	12	11	10	9	8
UBLEN							

7	6	5	4	3	2	1	0
UBLEN							

- **UBLEN: Channel x Microblock Length**

This field indicates the number of data in the microblock. The microblock contains UBLEN data.

Downloaded from Arrow.com.

35.9.27 XDMAC Channel x [x = 0..23] Block Control Register

Name: XDMAC_CBCx [x = 0..23]

Address: 0x40078074 [0], 0x400780B4 [1], 0x400780F4 [2], 0x40078134 [3], 0x40078174 [4], 0x400781B4 [5], 0x400781F4 [6], 0x40078234 [7], 0x40078274 [8], 0x400782B4 [9], 0x400782F4 [10], 0x40078334 [11], 0x40078374 [12], 0x400783B4 [13], 0x400783F4 [14], 0x40078434 [15], 0x40078474 [16], 0x400784B4 [17], 0x400784F4 [18], 0x40078534 [19], 0x40078574 [20], 0x400785B4 [21], 0x400785F4 [22], 0x40078634 [23]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	BLEN			

7	6	5	4	3	2	1	0
BLEN							

- **BLEN: Channel x Block Length**

The length of the block is (BLEN+1) microblocks.

Downloaded from Arrow.com.

35.9.28 XDMAC Channel x [x = 0..23] Configuration Register

Name: XDMAC_CCx[x = 0..23]

Address: 0x40078078 [0], 0x400780B8 [1], 0x400780F8 [2], 0x40078138 [3], 0x40078178 [4], 0x400781B8 [5], 0x400781F8 [6], 0x40078238 [7], 0x40078278 [8], 0x400782B8 [9], 0x400782F8 [10], 0x40078338 [11], 0x40078378 [12], 0x400783B8 [13], 0x400783F8 [14], 0x40078438 [15], 0x40078478 [16], 0x400784B8 [17], 0x400784F8 [18], 0x40078538 [19], 0x40078578 [20], 0x400785B8 [21], 0x400785F8 [22], 0x40078638 [23]

Access: Read/Write

31	30	29	28	27	26	25	24
–	PERID						

23	22	21	20	19	18	17	16
WRIP	RDIP	INITD	–	DAM		SAM	

15	14	13	12	11	10	9	8
–	DIF	SIF	DWIDTH		CSIZE		

7	6	5	4	3	2	1	0
MEMSET	SWREQ	–	DSYNC	–	MBSIZE		TYPE

- **TYPE: Channel x Transfer Type**

0 (MEM_TRAN): Self triggered mode (Memory to Memory Transfer).

1 (PER_TRAN): Synchronized mode (Peripheral to Memory or Memory to Peripheral Transfer).

- **MBSIZE: Channel x Memory Burst Size**

Value	Name	Description
0	SINGLE	The memory burst size is set to one.
1	FOUR	The memory burst size is set to four.
2	EIGHT	The memory burst size is set to eight.
3	SIXTEEN	The memory burst size is set to sixteen.

- **DSYNC: Channel x Synchronization**

0 (PER2MEM): Peripheral to Memory transfer.

1 (MEM2PER): Memory to Peripheral transfer.

- **SWREQ: Channel x Software Request Trigger**

0 (HWR_CONNECTED): Hardware request line is connected to the peripheral request line.

1 (SWR_CONNECTED): Software request is connected to the peripheral request line.

- **MEMSET: Channel x Fill Block of memory**

0 (NORMAL_MODE): Memset is not activated.

1 (HW_MODE): Sets the block of memory pointed by DA field to the specified value. This operation is performed on 8, 16 or 32 bits basis.

Downloaded from Arrow.com.

- **CSIZE: Channel x Chunk Size**

Value	Name	Description
0	CHK_1	1 data transferred
1	CHK_2	2 data transferred
2	CHK_4	4 data transferred
3	CHK_8	8 data transferred
4	CHK_16	16 data transferred

- **DWIDTH: Channel x Data Width**

Value	Name	Description
0	BYTE	The data size is set to 8 bits
1	HALFWORD	The data size is set to 16 bits
2	WORD	The data size is set to 32 bits

- **SIF: Channel x Source Interface Identifier**

0 (AHB_IF0): The data is read through the system bus interface 0.

1 (AHB_IF1): The data is read through the system bus interface 1.

- **DIF: Channel x Destination Interface Identifier**

0 (AHB_IF0): The data is written through the system bus interface 0.

1 (AHB_IF1): The data is written though the system bus interface 1.

- **SAM: Channel x Source Addressing Mode**

Value	Name	Description
0	FIXED_AM	The address remains unchanged.
1	INCREMENTED_AM	The addressing mode is incremented (the increment size is set to the data size).
2	UBS_AM	The microblock stride is added at the microblock boundary.
3	UBS_DS_AM	The microblock stride is added at the microblock boundary, the data stride is added at the data boundary.

- **DAM: Channel x Destination Addressing Mode**

Value	Name	Description
0	FIXED_AM	The address remains unchanged.
1	INCREMENTED_AM	The addressing mode is incremented (the increment size is set to the data size).
2	UBS_AM	The microblock stride is added at the microblock boundary.
3	UBS_DS_AM	The microblock stride is added at the microblock boundary, the data stride is added at the data boundary.

- **INITD: Channel Initialization Done (this bit is read-only)**

0 (IN_PROGRESS): Channel initialization is in progress.

1 (TERMINATED): Channel initialization is completed.

Downloaded from Arrow.com.

- **RDIP: Read in Progress (this bit is read-only)**

0 (DONE): No Active read transaction on the bus.

1 (IN_PROGRESS): A read transaction is in progress.

- **WRIP: Write in Progress (this bit is read-only)**

0 (DONE): No Active write transaction on the bus.

1 (IN_PROGRESS): A Write transaction is in progress.

- **PERID: Channel x Peripheral Hardware Request Line Identifier**

This field contains the peripheral hardware request line identifier. PERID refers to identifiers defined in Section 35.4 ”DMA Controller Peripheral Connections”.

Downloaded from Arrow.com.

35.9.29 XDMAC Channel x [x = 0..23] Data Stride Memory Set Pattern Register

Name: XDMAC_CDS_MSPx [x = 0..23]

Address: 0x4007807C [0], 0x400780BC [1], 0x400780FC [2], 0x4007813C [3], 0x4007817C [4], 0x400781BC [5], 0x400781FC [6], 0x4007823C [7], 0x4007827C [8], 0x400782BC [9], 0x400782FC [10], 0x4007833C [11], 0x4007837C [12], 0x400783BC [13], 0x400783FC [14], 0x4007843C [15], 0x4007847C [16], 0x400784BC [17], 0x400784FC [18], 0x4007853C [19], 0x4007857C [20], 0x400785BC [21], 0x400785FC [22], 0x4007863C [23]

Access: Read/Write

31	30	29	28	27	26	25	24
DDS_MSP							
23	22	21	20	19	18	17	16
DDS_MSP							
15	14	13	12	11	10	9	8
SDS_MSP							
7	6	5	4	3	2	1	0
SDS_MSP							

- **SDS_MSP: Channel x Source Data stride or Memory Set Pattern**

When XDMAC_CCx.MEMSET = 0, this field indicates the source data stride.

When XDMAC_CCx.MEMSET = 1, this field indicates the memory set pattern.

- **DDS_MSP: Channel x Destination Data Stride or Memory Set Pattern**

When XDMAC_CCx.MEMSET = 0, this field indicates the destination data stride.

When XDMAC_CCx.MEMSET = 1, this field indicates the memory set pattern.

Downloaded from Arrow.com.

35.9.30 XDMAC Channel x [x = 0..23] Source Microblock Stride Register

Name: XDMAC_CSUSx [x = 0..23]

Address: 0x40078080 [0], 0x400780C0 [1], 0x40078100 [2], 0x40078140 [3], 0x40078180 [4], 0x400781C0 [5], 0x40078200 [6], 0x40078240 [7], 0x40078280 [8], 0x400782C0 [9], 0x40078300 [10], 0x40078340 [11], 0x40078380 [12], 0x400783C0 [13], 0x40078400 [14], 0x40078440 [15], 0x40078480 [16], 0x400784C0 [17], 0x40078500 [18], 0x40078540 [19], 0x40078580 [20], 0x400785C0 [21], 0x40078600 [22], 0x40078640 [23]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
SUBS							

15	14	13	12	11	10	9	8
SUBS							

7	6	5	4	3	2	1	0
SUBS							

- **SUBS: Channel x Source Microblock Stride**

Two’s complement microblock stride for channel x.

Downloaded from Arrow.com.

35.9.31 XDMAC Channel x [x = 0..23] Destination Microblock Stride Register

Name: XDMAC_CDUSx [x = 0..23]

Address: 0x40078084 [0], 0x400780C4 [1], 0x40078104 [2], 0x40078144 [3], 0x40078184 [4], 0x400781C4 [5], 0x40078204 [6], 0x40078244 [7], 0x40078284 [8], 0x400782C4 [9], 0x40078304 [10], 0x40078344 [11], 0x40078384 [12], 0x400783C4 [13], 0x40078404 [14], 0x40078444 [15], 0x40078484 [16], 0x400784C4 [17], 0x40078504 [18], 0x40078544 [19], 0x40078584 [20], 0x400785C4 [21], 0x40078604 [22], 0x40078644 [23]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
DUBS							

15	14	13	12	11	10	9	8
DUBS							

7	6	5	4	3	2	1	0
DUBS							

- **DUBS: Channel x Destination Microblock Stride**

Two’s complement microblock stride for channel x.

Downloaded from Arrow.com.

36. Image Sensor Interface (ISI)

36.1 Description

The Image Sensor Interface (ISI) connects a CMOS-type image sensor to the processor and provides image capture in various formats.The ISI performs data conversion, if necessary, before the storage in memory through DMA.

The ISI supports color CMOS image sensor and grayscale image sensors with a reduced set of functionalities.

In grayscale mode, the data stream is stored in memory without any processing and so is not compatible with the LCD controller.

Internal FIFOs on the preview and codec paths are used to store the incoming data. The RGB output on the preview path is compatible with the LCD controller. This module outputs the data in RGB format (LCD compatible) and has scaling capabilities to make it compliant to the LCD display resolution (see Table 36-5 on page 530).

Several input formats such as preprocessed RGB or YCbCr are supported through the data bus interface.

The ISI supports two modes of synchronization:

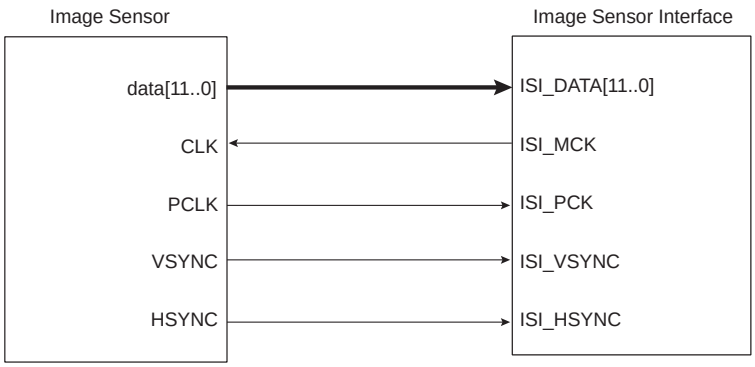
- Hardware with ISI_VSYNC and ISI_HSYNC signals
- International Telecommunication Union Recommendation *ITU-R BT.656-4* Start-of-Active-Video (SAV) and End-of-Active-Video (EAV) synchronization sequence

Using EAV/SAV for synchronization reduces the pin count (ISI_VSYNC, ISI_HSYNC not used). The polarity of the synchronization pulse is programmable to comply with the sensor signals.

Table 36-1. I/O Description

Signal	Direction	Description
ISI_VSYNC	In	Vertical Synchronization
ISI_HSYNC	In	Horizontal Synchronization
ISI_DATA[11..0]	In	Sensor Pixel Data
ISI_MCK	Out	Master Clock Provided to the Image Sensor
ISI_PCK	In	Pixel Clock Provided by the Image Sensor

Figure 36-1. ISI Connection Example



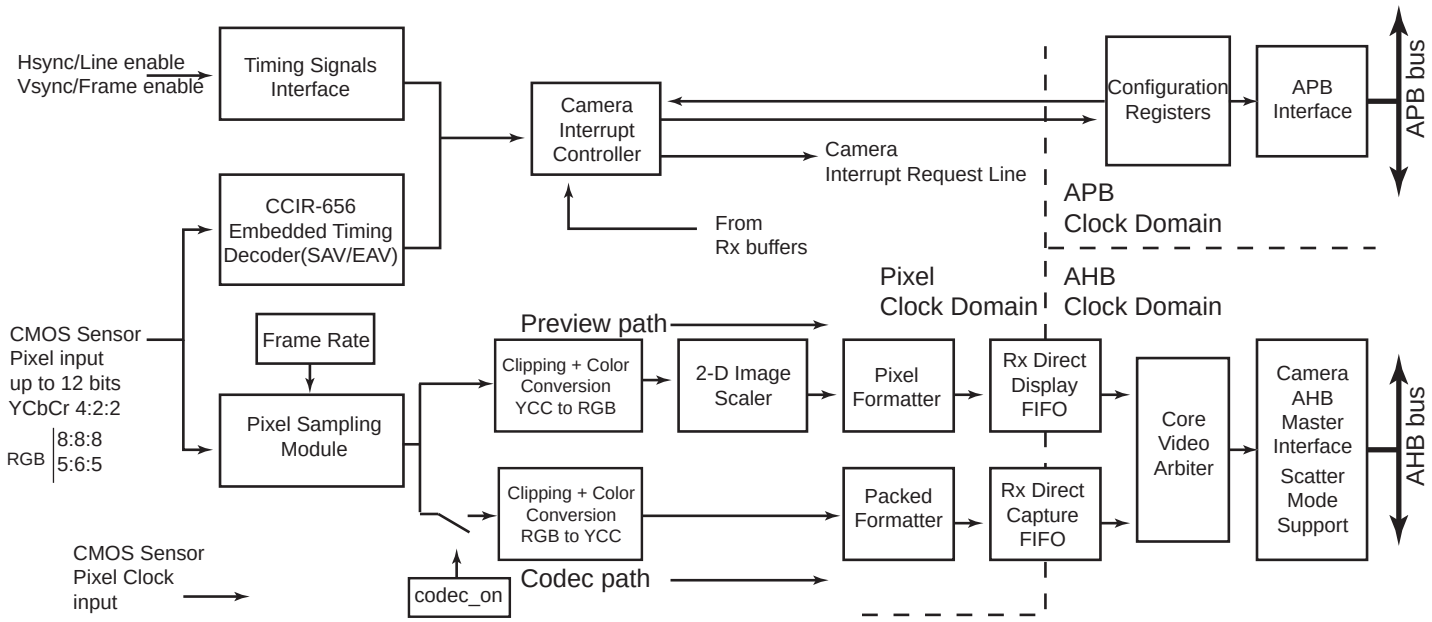
Downloaded from Arrow.com.

36.2 Embedded Characteristics

- ITU-R BT. 601/656 8-bit Mode External Interface Support
- Supports up to 12-bit Grayscale CMOS Sensors
- Support for ITU-R BT.656-4 SAV and EAV Synchronization
- Vertical and Horizontal Resolutions up to 2048 × 2048
- Preview Path up to 640 × 480 in RGB Mode
- Codec Path up to 2048 × 2048
- 16-byte FIFO on Codec Path
- 16-byte FIFO on Preview Path
- Support for Packed Data Formatting for YCbCr 4:2:2 Formats
- Preview Scaler to Generate Smaller Size image
- Programmable Frame Capture Rate
- VGA, QVGA, CIF, QCIF Formats Supported for LCD Preview
- Custom Formats with Horizontal and Vertical Preview Size as Multiples of 16 Also Supported for LCD Preview

36.3 Block Diagram

Figure 36-2. ISI Block Diagram



Downloaded from Arrow.com.

36.4 Product Dependencies

36.4.1 I/O Lines

The pins used for interfacing the compliant external devices can be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the ISI pins to their peripheral functions.

Table 36-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
ISI	ISI_D0	PD22	D
ISI	ISI_D1	PD21	D
ISI	ISI_D2	PB3	D
ISI	ISI_D3	PA9	B
ISI	ISI_D4	PA5	B
ISI	ISI_D5	PD11	D
ISI	ISI_D6	PD12	D
ISI	ISI_D7	PA27	D
ISI	ISI_D8	PD27	D
ISI	ISI_D9	PD28	D
ISI	ISI_D10	PD30	D
ISI	ISI_D11	PD31	D
ISI	ISI_HSYNC	PD24	D
ISI	ISI_PCK	PA24	D
ISI	ISI_VSYNC	PD25	D

36.4.2 Power Management

The ISI can be clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the ISI clock.

36.4.3 Interrupt Sources

The ISI interface has an interrupt line connected to the interrupt controller. Handling the ISI interrupt requires programming the interrupt controller before configuring the ISI.

Table 36-3. Peripheral IDs

Instance	ID
ISI	59

Downloaded from Arrow.com.

36.5 Functional Description

The Image Sensor Interface (ISI) supports direct connection to the ITU-R BT. 601/656 8-bit mode compliant sensors and up to 12-bit grayscale sensors. It receives the image data stream from the image sensor on the 12-bit data bus.

This module receives up to 12 bits for data, the horizontal and vertical synchronizations and the pixel clock. The reduced pin count alternative for synchronization is supported for sensors that embed SAV (start of active video) and EAV (end of active video) delimiters in the data stream.

The Image Sensor Interface interrupt line is connected to the Advanced Interrupt Controller and can trigger an interrupt at the beginning of each frame and at the end of a DMA frame transfer. If the SAV/EAV synchronization is used, an interrupt can be triggered on each delimiter event.

For 8-bit color sensors, the data stream received can be in several possible formats: YCbCr 4:2:2, RGB 8:8:8, RGB 5:6:5 and may be processed before the storage in memory. When the preview DMA channel is configured and enabled, the preview path is activated and an ‘RGB frame’ is moved to memory. The preview path frame rate is configured with the FRATE field of the ISI_CFG1 register. When the codec DMA channel is configured and enabled, the codec path is activated and a ‘YCbCr 4:2:2 frame’ is captured as soon as the ISI_CDC bit of the ISI Control Register (ISI_CR) is set.

When the FULL bit of the ISI_CFG1 register is set, both preview DMA channel and codec DMA channel can operate simultaneously. When a zero is written to the FULL bit of the ISI_CFG1 register, a hardware scheduler checks the FRATE field. If its value is zero, a preview frame is skipped and a codec frame is moved to memory instead. If its value is other than zero, at least one free frame slot is available. The scheduler postpones the codec frame to that free available frame slot.

The data stream may be sent on both preview path and codec path if the value of bit ISI_CDC in the ISI_CR is one. To optimize the bandwidth, the codec path should be enabled only when a capture is required.

In grayscale mode, the input data stream is stored in memory without any processing. The 12-bit data, which represent the grayscale level for the pixel, is stored in memory one or two pixels per word, depending on the GS_MODE bit in the ISI_CFG2 register. The codec datapath is not available when grayscale image is selected.

A frame rate counter allows users to capture all frames or 1 out of every 2 to 8 frames.

Downloaded from Arrow.com.

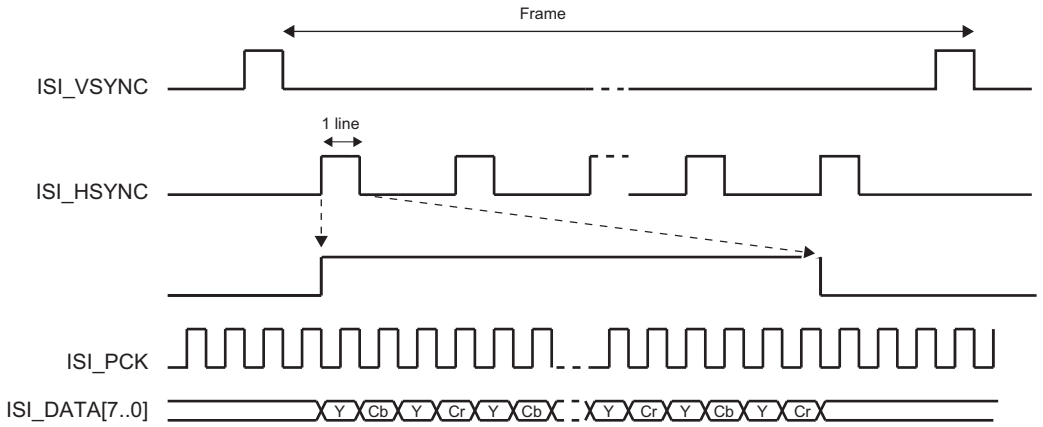
36.5.1 Data Timing

36.5.1.1 VSYNC/HSYNC Data Timing

In the VSYNC/HSYNC synchronization, the valid data is captured with the active edge of the pixel clock (ISI_PCK), after SFD lines of vertical blanking and SLD pixel clock periods delay programmed in the ISI_CR.

The data timing using horizontal and vertical synchronization are shown in Figure 36-4.

Figure 36-3. HSYNC and VSYNC Synchronization



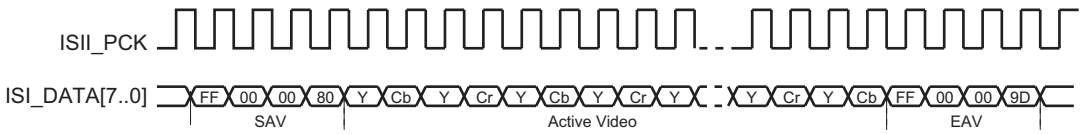
36.5.1.2 SAV/EAV Data Timing

The ITU-RBT.656-4 standard defines the functional timing for an 8-bit wide interface.

There are two timing reference signals, one at the beginning of each video data block SAV (0xFF000080) and one at the end of each video data block EAV (0xFF00009D). Only data sent between EAV and SAV is captured. Horizontal blanking and vertical blanking are ignored. Use of the SAV and EAV synchronization eliminates the ISI_VSYNC and ISI_HSYNC signals from the interface, thereby reducing the pin count. In order to retrieve both frame and line synchronization properly, at least one line of vertical blanking is mandatory.

The data timing using EAV/SAV sequence synchronization are shown in Figure 36-4.

Figure 36-4. SAV and EAV Sequence Synchronization



Downloaded from Arrow.com.

36.5.2 Data Ordering

The RGB color space format is required for viewing images on a display screen preview, and the YCbCr color space format is required for encoding.

All the sensors do not output the YCbCr or RGB components in the same order. The ISI allows the user to program the same component order as the sensor, reducing software treatments to restore the right format.

Table 36-4. Data Ordering in YCbCr Mode

Mode	Byte 0	Byte 1	Byte 2	Byte 3
Default	Cb(i)	Y(i)	Cr(i)	Y(i+1)
Mode 1	Cr(i)	Y(i)	Cb(i)	Y(i+1)
Mode 2	Y(i)	Cb(i)	Y(i+1)	Cr(i)
Mode 3	Y(i)	Cr(i)	Y(i+1)	Cb(i)

Table 36-5. RGB Format in Default Mode, RGB_CFG = 00, No Swap

Mode	Byte	D7	D6	D5	D4	D3	D2	D1	D0
RGB 8:8:8	Byte 0	R7(i)	R6(i)	R5(i)	R4(i)	R3(i)	R2(i)	R1(i)	R0(i)
	Byte 1	G7(i)	G6(i)	G5(i)	G4(i)	G3(i)	G2(i)	G1(i)	G0(i)
	Byte 2	B7(i)	B6(i)	B5(i)	B4(i)	B3(i)	B2(i)	B1(i)	B0(i)
	Byte 3	R7(i+1)	R6(i+1)	R5(i+1)	R4(i+1)	R3(i+1)	R2(i+1)	R1(i+1)	R0(i+1)
RGB 5:6:5	Byte 0	R4(i)	R3(i)	R2(i)	R1(i)	R0(i)	G5(i)	G4(i)	G3(i)
	Byte 1	G2(i)	G1(i)	G0(i)	B4(i)	B3(i)	B2(i)	B1(i)	B0(i)
	Byte 2	R4(i+1)	R3(i+1)	R2(i+1)	R1(i+1)	R0(i+1)	G5(i+1)	G4(i+1)	G3(i+1)
	Byte 3	G2(i+1)	G1(i+1)	G0(i+1)	B4(i+1)	B3(i+1)	B2(i+1)	B1(i+1)	B0(i+1)

Table 36-6. RGB Format, RGB_CFG = 10 (Mode 2), No Swap

Mode	Byte	D7	D6	D5	D4	D3	D2	D1	D0
RGB 5:6:5	Byte 0	G2(i)	G1(i)	G0(i)	R4(i)	R3(i)	R2(i)	R1(i)	R0(i)
	Byte 1	B4(i)	B3(i)	B2(i)	B1(i)	B0(i)	G5(i)	G4(i)	G3(i)
	Byte 2	G2(i+1)	G1(i+1)	G0(i+1)	R4(i+1)	R3(i+1)	R2(i+1)	R1(i+1)	R0(i+1)
	Byte 3	B4(i+1)	B3(i+1)	B2(i+1)	B1(i+1)	B0(i+1)	G5(i+1)	G4(i+1)	G3(i+1)

Table 36-7. RGB Format in Default Mode, RGB_CFG = 00, Swap Activated

Mode	Byte	D7	D6	D5	D4	D3	D2	D1	D0
RGB 8:8:8	Byte 0	R0(i)	R1(i)	R2(i)	R3(i)	R4(i)	R5(i)	R6(i)	R7(i)
	Byte 1	G0(i)	G1(i)	G2(i)	G3(i)	G4(i)	G5(i)	G6(i)	G7(i)
	Byte 2	B0(i)	B1(i)	B2(i)	B3(i)	B4(i)	B5(i)	B6(i)	B7(i)
	Byte 3	R0(i+1)	R1(i+1)	R2(i+1)	R3(i+1)	R4(i+1)	R5(i+1)	R6(i+1)	R7(i+1)
RGB 5:6:5	Byte 0	G3(i)	G4(i)	G5(i)	R0(i)	R1(i)	R2(i)	R3(i)	R4(i)
	Byte 1	B0(i)	B1(i)	B2(i)	B3(i)	B4(i)	G0(i)	G1(i)	G2(i)
	Byte 2	G3(i+1)	G4(i+1)	G5(i+1)	R0(i+1)	R1(i+1)	R2(i+1)	R3(i+1)	R4(i+1)
	Byte 3	B0(i+1)	B1(i+1)	B2(i+1)	B3(i+1)	B4(i+1)	G0(i+1)	G1(i+1)	G2(i+1)

Downloaded from Arrow.com.

The RGB 5:6:5 input format is processed to be displayed as RGB 5:6:5 format, compliant with the 16-bit mode of the LCD controller.

36.5.3 Clocks

The sensor master clock (ISI_MCK) can be generated either by the Advanced Power Management Controller (APMC) through a Programmable Clock output or by an external oscillator connected to the sensor.

None of the sensors embed a power management controller, so providing the clock by the APMC is a simple and efficient way to control power consumption of the system.

Care must be taken when programming the system clock. The ISI has two clock domains, the sensor master clock and the pixel clock provided by sensor. The two clock domains are not synchronized, but the sensor master clock must be faster than the pixel clock.

36.5.4 Preview Path

36.5.4.1 Scaling, Decimation (Subsampling)

This module resizes captured 8-bit color sensor images to fit the LCD display format. The resize module performs only downscaling. The same ratio is applied for both horizontal and vertical resize, then a fractional decimation algorithm is applied.

The decimation factor is a multiple of 1/16; values 0 to 15 are forbidden.

Table 36-8. Decimation Factor

Decimation Value	0–15	16	17	18	19	...	124	125	126	127
Decimation Factor	—	1	1.063	1.125	1.188	...	7.750	7.813	7.875	7.938

Table 36-9. Decimation and Scaler Offset Values

OUTPUT	INPUT	352 × 288	640 × 480	800 × 600	1280 × 1024	1600 × 1200	2048 × 1536
VGA 640 × 480	F	—	16	20	32	40	51
QVGA 320 × 240	F	16	32	40	64	80	102
CIF 352 × 288	F	16	26	33	56	66	85
QCIF 176 × 144	F	32	53	66	113	133	170

Example:

Input 1280 × 1024 Output = 640 × 480

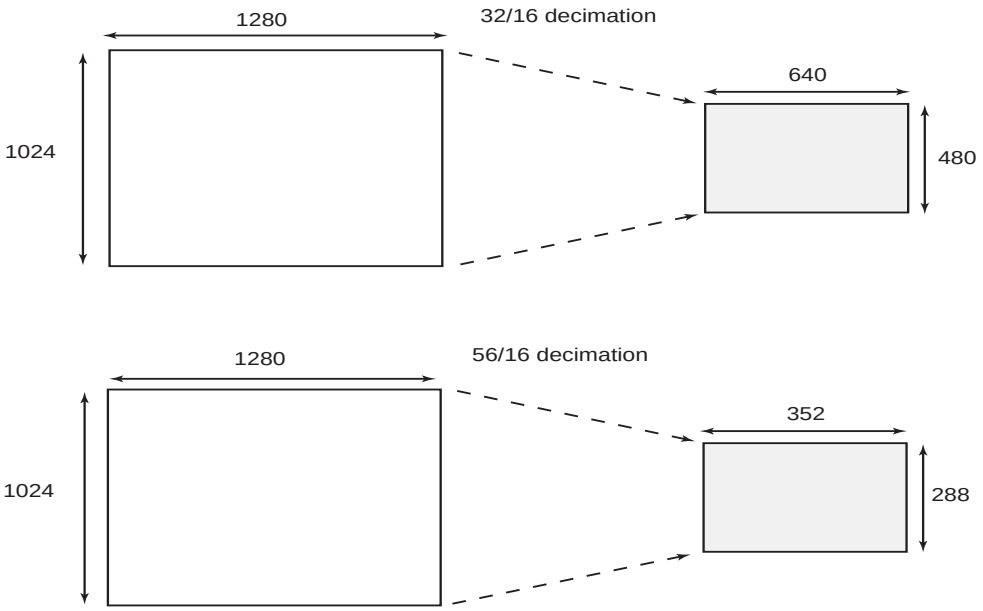
Hratio = 1280/640 = 2

Vratio = 1024/480 = 2.1333

The decimation factor is 2 so 32/16.

Downloaded from Arrow.com.

Figure 36-5. Resize Examples



36.5.4.2 Color Space Conversion

This module converts YCrCb or YUV pixels to RGB color space. Clipping is performed to ensure that the samples value do not exceed the allowable range. The conversion matrix is defined below and is fully programmable:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} C_0 & 0 & C_1 \\ C_0 & -C_2 & -C_3 \\ C_0 & C_4 & 0 \end{bmatrix} \times \begin{bmatrix} Y - Y_{off} \\ C_b - C_{boff} \\ C_r - C_{roff} \end{bmatrix}$$

Example of programmable value to convert YCrCb to RGB:

$$\begin{cases} R = 1.164 \cdot (Y - 16) + 1.596 \cdot (C_r - 128) \\ G = 1.164 \cdot (Y - 16) - 0.813 \cdot (C_r - 128) - 0.392 \cdot (C_b - 128) \\ B = 1.164 \cdot (Y - 16) + 2.107 \cdot (C_b - 128) \end{cases}$$

An example of programmable value to convert from YUV to RGB:

$$\begin{cases} R = Y + 1.596 \cdot V \\ G = Y - 0.394 \cdot U - 0.436 \cdot V \\ B = Y + 2.032 \cdot U \end{cases}$$

Downloaded from Arrow.com.

36.5.4.3 Memory Interface

RGB Mode

The preview datapath contains a data formatter that converts 8:8:8 pixel to RGB 5:6:5 format compliant with the 16-bit format of the LCD controller. In general, when converting from a color channel with more bits to one with fewer bits, the formatter module discards the lower-order bits.

For example, converting from RGB 8:8:8 to RGB 5:6:5, the formatter module discards the three LSBs from the red and blue channels, and two LSBs from the green channel.

12-bit Grayscale Mode

ISI_DATA[11:0] is the physical interface to the ISI. These bits are sampled and written to memory.

When 12-bit grayscale mode is enabled, two memory formats are supported:

ISI_CFG2.GS_MODE = 0: two pixels per word

ISI_CFG2.GS_MODE = 1: one pixel per word

The following tables illustrate the memory mapping for the two formats.

Table 36-10. Grayscale Memory Mapping Configuration for 12-bit Data (ISI_CFG2.GS_MODE = 0: two pixels per word)

31	30	29	28	27	26	25	24
Pixel 0 [11:4]							
23	22	21	20	19	18	17	16
Pixel 0 [3:0]				–	–	–	–
15	14	13	12	11	10	9	8
Pixel 1 [11:4]							
7	6	5	4	3	2	1	0
Pixel 1 [3:0]				–	–	–	–

Table 36-11. Grayscale Memory Mapping Configuration for 12-bit Data (ISI_CFG2.GS_MODE = 1: one pixel per word)

31	30	29	28	27	26	25	24
Pixel 0 [11:4]							
23	22	21	20	19	18	17	16
Pixel 0 [3:0]				–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

Downloaded from Arrow.com.

8-bit Grayscale Mode

For 8-bit grayscale mode, ISI_DATA[7:0] on the 12-bit data bus is the physical interface to the ISI. These bits are sampled and written to memory.

To enable 8-bit grayscale mode, configure ISI_CFG2 as follows:

- Clear ISI_CFG2.GRAYSCALE.
- Clear ISI_CFG2.RGB_SWAP.
- Clear ISI_CFG2.COL_SPACE.
- Configure the field ISI_CFG2.YCC_SWAP to value 0.
- Configure the field ISI_CFG2.IM_VSIZE with the vertical resoloution of the image minus 1.
- Configure the field ISI_CFG2.IM_HSIZE with the horizontal resolution of the image divided by 2. The horizontal resolution must be a multiple of 2.

The codec datapath is used to capture the 8-bit grayscale image. Use the following configuration:

- Set ISI_DMA_C_CTRL.C_FETCH.
- Configure ISI_DMA_C_DSCR.C_DSCR with the descriptor address.
- Write a one to the bit ISI_DMA_CHER.C_CH_EN.

Table 36-12. Memory Mapping for 8-bit Grayscale Mode

31	30	29	28	27	26	25	24
Pixel 3							
23	22	21	20	19	18	17	16
Pixel 2							
15	14	13	12	11	10	9	8
Pixel 1							
7	6	5	4	3	2	1	0
Pixel 0							

Downloaded from Arrow.com.

36.5.4.4 FIFO and DMA Features

Both preview and codec datapaths contain FIFOs. These asynchronous buffers are used to safely transfer formatted pixels from the pixel clock domain to the AHB clock domain. A video arbiter is used to manage FIFO thresholds and triggers a relevant DMA request through the AHB master interface. Thus, depending on the FIFO state, a specified length burst is asserted. Regarding AHB master interface, it supports Scatter DMA mode through linked list operation. This mode of operation improves flexibility of image buffer location and allows the user to allocate two or more frame buffers. The destination frame buffers are defined by a series of Frame Buffer Descriptors (FBD). Each FBD controls the transfer of one entire frame and then optionally loads a further FBD to switch the DMA operation at another frame buffer address. The FBD is defined by a series of three words. The first one defines the current frame buffer address (named DMA_X_ADDR register), the second defines control information (named DMA_X_CTRL register) and the third defines the next descriptor address (named DMA_X_DSCR). DMA transfer mode with linked list support is available for both codec and preview datapath. The data to be transferred described by an FBD requires several burst accesses. In the following example, the use of two ping-pong frame buffers is described.

Example:

The first FBD, stored at address 0x00030000, defines the location of the first frame buffer. This address is programmed in the ISI user interface DMA_P_DSCR. To enable the descriptor fetch operation, the value 0x00000001 must be written to the DMA_P_CTRL register. LLI_0 and LLI_1 are the two descriptors of the linked list.

Destination address: frame buffer ID0 0x02A000 (LLI_0.DMA_P_ADDR)

Transfer 0 Control Information, fetch and writeback: 0x00000003 (LLI_0.DMA_P_CTRL)

Next FBD address: 0x00030010 (LLI_0.DMA_P_DSCR)

Second FBD, stored at address 0x00030010, defines the location of the second frame buffer.

Destination address: frame buffer ID1 0x0003A000 (LLI_1.DMA_P_ADDR)

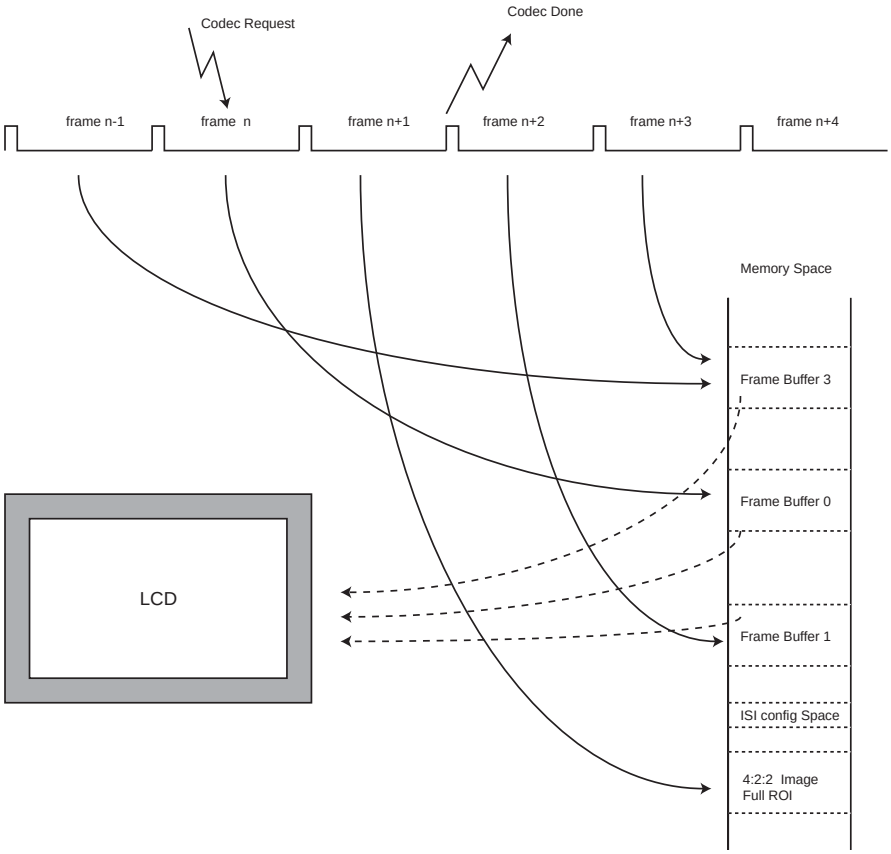
Transfer 1 Control information fetch and writeback: 0x00000003 (LLI_1.DMA_P_CTRL)

Next FBD address: 0x00030000, wrapping to first FBD (LLI_1.DMA_P_DSCR)

Using this technique, several frame buffers can be configured through the linked list. Figure 36-6 illustrates a typical three frame buffer application. Frame n is mapped to frame buffer 0, frame n+1 is mapped to frame buffer 1, frame n+2 is mapped to frame buffer 2, further frames wrap. A codec request occurs, and the full-size 4:2:2 encoded frame is stored in a dedicated memory space.

Downloaded from Arrow.com.

Figure 36-6. Three Frame Buffers Application and Memory Mapping



Downloaded from Arrow.com.

36.5.5 Codec Path

36.5.5.1 Color Space Conversion

Depending on user selection, this module can be bypassed so that input YCrCb stream is directly connected to the format converter module. If the RGB input stream is selected, this module converts RGB to YCrCb color space with the formulas given below:

$$\begin{bmatrix} Y \\ C_r \\ C_b \end{bmatrix} = \begin{bmatrix} C_0 & C_1 & C_2 \\ C_3 & -C_4 & -C_5 \\ -C_6 & -C_7 & C_8 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} Y_{off} \\ Cr_{off} \\ Cb_{off} \end{bmatrix}$$

An example of coefficients is given below:

$$\begin{cases} Y = 0.257 \cdot R + 0.504 \cdot G + 0.098 \cdot B + 16 \\ C_r = 0.439 \cdot R - 0.368 \cdot G - 0.071 \cdot B + 128 \\ C_b = -0.148 \cdot R - 0.291 \cdot G + 0.439 \cdot B + 128 \end{cases}$$

36.5.5.2 Memory Interface

Dedicated FIFOs are used to support packed memory mapping. YCrCb pixel components are sent in a single 32-bit word in a contiguous space (packed). Data is stored in the order of natural scan lines. Planar mode is not supported.

36.5.5.3 DMA Features

Like preview datapath, codec datapath DMA mode uses linked list operation.

Downloaded from Arrow.com.

36.6 Image Sensor Interface (ISI) User Interface

Table 36-13. Register Mapping

Offset	Register	Name	Access	Reset Value
0x00	ISI Configuration 1 Register	ISI_CFG1	Read/Write	0x00000000
0x04	ISI Configuration 2 Register	ISI_CFG2	Read/Write	0x00000000
0x08	ISI Preview Size Register	ISI_PSIZE	Read/Write	0x00000000
0x0C	ISI Preview Decimation Factor Register	ISI_PDECF	Read/Write	0x00000010
0x10	ISI Color Space Conversion YCrCb To RGB Set 0 Register	ISI_Y2R_SET0	Read/Write	0x6832CC95
0x14	ISI Color Space Conversion YCrCb To RGB Set 1 Register	ISI_Y2R_SET1	Read/Write	0x00007102
0x18	ISI Color Space Conversion RGB To YCrCb Set 0 Register	ISI_R2Y_SET0	Read/Write	0x01324145
0x1C	ISI Color Space Conversion RGB To YCrCb Set 1 Register	ISI_R2Y_SET1	Read/Write	0x01245E38
0x20	ISI Color Space Conversion RGB To YCrCb Set 2 Register	ISI_R2Y_SET2	Read/Write	0x01384A4B
0x24	ISI Control Register	ISI_CR	Write-only	–
0x28	ISI Status Register	ISI_SR	Read-only	0x00000000
0x2C	ISI Interrupt Enable Register	ISI_IER	Write-only	–
0x30	ISI Interrupt Disable Register	ISI_IDR	Write-only	–
0x34	ISI Interrupt Mask Register	ISI_IMR	Read-only	0x00000000
0x38	DMA Channel Enable Register	ISI_DMA_CHER	Write-only	–
0x3C	DMA Channel Disable Register	ISI_DMA_CHDR	Write-only	–
0x40	DMA Channel Status Register	ISI_DMA_CHSR	Read-only	0x00000000
0x44	DMA Preview Base Address Register	ISI_DMA_P_ADDR	Read/Write	0x00000000
0x48	DMA Preview Control Register	ISI_DMA_P_CTRL	Read/Write	0x00000000
0x4C	DMA Preview Descriptor Address Register	ISI_DMA_P_DSCR	Read/Write	0x00000000
0x50	DMA Codec Base Address Register	ISI_DMA_C_ADDR	Read/Write	0x00000000
0x54	DMA Codec Control Register	ISI_DMA_C_CTRL	Read/Write	0x00000000
0x58	DMA Codec Descriptor Address Register	ISI_DMA_C_DSCR	Read/Write	0x00000000
0x5C–0xE0	Reserved	–	–	–
0xE4	Write Protection Mode Register	ISI_WPMR	Read/Write	0x00000000
0xE8	Write Protection Status Register	ISI_WPSR	Read-only	0x00000000
0xEC–0xF8	Reserved	–	–	–
0xFC	Reserved	–	–	–

Note: Several parts of the ISI controller use the pixel clock provided by the image sensor (ISI_PCK). Thus the user must first program the image sensor to provide this clock (ISI_PCK) before programming the Image Sensor Controller.

Downloaded from Arrow.com.

36.6.1 ISI Configuration 1 Register

Name: ISI_CFG1

Address: 0x4004C000

Access: Read/Write

31	30	29	28	27	26	25	24
SFD							

23	22	21	20	19	18	17	16
SLD							

15	14	13	12	11	10	9	8
–	THMASK		FULL	DISCR	FRATE		

7	6	5	4	3	2	1	0
CRC_SYNC	EMB_SYNC	–	PIXCLK_POL	VSYNC_POL	HSYNC_POL	–	–

- **HSYNC_POL: Horizontal Synchronization Polarity**

0: HSYNC active high.

1: HSYNC active low.

- **VSYNC_POL: Vertical Synchronization Polarity**

0: VSYNC active high.

1: VSYNC active low.

- **PIXCLK_POL: Pixel Clock Polarity**

0: Data is sampled on rising edge of pixel clock.

1: Data is sampled on falling edge of pixel clock.

- **EMB_SYNC: Embedded Synchronization**

0: Synchronization by HSYNC, VSYNC.

1: Synchronization by embedded synchronization sequence SAV/EAV.

- **CRC_SYNC: Embedded Synchronization Correction**

0: No CRC correction is performed on embedded synchronization.

1: CRC correction is performed. If the correction is not possible, the current frame is discarded and the CRC_ERR bit is set in the ISI_SR.

- **FRATE: Frame Rate [0..7]**

0: All the frames are captured, else one frame every FRATE + 1 is captured.

- **DISCR: Disable Codec Request**

0: Codec datapath DMA interface requires a request to restart.

1: Codec datapath DMA automatically restarts.

Downloaded from Arrow.com.

- **FULL: Full Mode is Allowed**

0: The codec frame is transferred to memory when an available frame slot is detected.

1: Both preview and codec DMA channels are operating simultaneously.

- **THMASK: Threshold Mask**

Value	Name	Description
0	BEATS_4	Only 4 beats AHB burst allowed
1	BEATS_8	Only 4 and 8 beats AHB burst allowed
2	BEATS_16	4, 8 and 16 beats AHB burst allowed

- **SLD: Start of Line Delay**

SLD pixel clock periods to wait before the beginning of a line.

- **SFD: Start of Frame Delay**

SFD lines are skipped at the beginning of the frame.

Downloaded from Arrow.com.

36.6.2 ISI Configuration 2 Register

Name: ISI_CFG2

Address: 0x4004C004

Access: Read/Write

31	30	29	28	27	26	25	24
RGB_CFG		YCC_SWAP		–	IM_HSIZE		
23	22	21	20	19	18	17	16
IM_HSIZE							
15	14	13	12	11	10	9	8
COL_SPACE	RGB_SWAP	GRAYSCALE	RGB_MODE	GS_MODE	IM_VSIZE		
7	6	5	4	3	2	1	0
IM_VSIZE							

- **IM_VSIZE: Vertical Size of the Image Sensor [0..2047]**

IM_VSIZE = Vertical size - 1

- **GS_MODE: Grayscale Pixel Format Mode**

0: 2 pixels per word.

1: 1 pixel per word.

- **RGB_MODE: RGB Input Mode**

0: RGB 8:8:8 24 bits.

1: RGB 5:6:5 16 bits.

- **GRAYSCALE: Grayscale Mode Format Enable**

0: Grayscale mode is disabled.

1: Input image is assumed to be grayscale-coded.

- **RGB_SWAP: RGB Format Swap Mode**

0: D7 → R7.

1: D0 → R7.

The RGB_SWAP has no effect when grayscale mode is enabled.

- **COL_SPACE: Color Space for the Image Data**

0: YCbCr.

1: RGB.

- **IM_HSIZE: Horizontal Size of the Image Sensor [0..2047]**

If 8-bit grayscale mode is enabled, IM_HSIZE = (Horizontal size/2) - 1.

Else IM_HSIZE = Horizontal size - 1.

Downloaded from Arrow.com.

- **YCC_SWAP: YCrCb Format Swap Mode**

Defines the YCC image data.

Value	Name	Description
0	DEFAULT	Byte 0 Cb(i) Byte 1 Y(i) Byte 2 Cr(i) Byte 3 Y(i+1)
1	MODE1	Byte 0 Cr(i) Byte 1 Y(i) Byte 2 Cb(i) Byte 3 Y(i+1)
2	MODE2	Byte 0 Y(i) Byte 1 Cb(i) Byte 2 Y(i+1) Byte 3 Cr(i)
3	MODE3	Byte 0 Y(i) Byte 1 Cr(i) Byte 2 Y(i+1) Byte 3 Cb(i)

- **RGB_CFG: RGB Pixel Mapping Configuration**

Defines RGB pattern when RGB_MODE is set to 1.

Value	Name	Description
0	DEFAULT	Byte 0 R/G(MSB) Byte 1 G(LSB)/B Byte 2 R/G(MSB) Byte 3 G(LSB)/B
1	MODE1	Byte 0 B/G(MSB) Byte 1 G(LSB)/R Byte 2 B/G(MSB) Byte 3 G(LSB)/R
2	MODE2	Byte 0 G(LSB)/R Byte 1 B/G(MSB) Byte 2 G(LSB)/R Byte 3 B/G(MSB)
3	MODE3	Byte 0 G(LSB)/B Byte 1 R/G(MSB) Byte 2 G(LSB)/B Byte 3 R/G(MSB)

If RGB_MODE is set to RGB 8:8:8, then RGB_CFG = 0 implies RGB color sequence, else it implies BGR color sequence.

Downloaded from Arrow.com.

36.6.3 ISI Preview Size Register

Name: ISI_PSIZE

Address: 0x4004C008

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	PREV_HSIZE	

23	22	21	20	19	18	17	16
PREV_HSIZE							

15	14	13	12	11	10	9	8
–	–	–	–	–	–	PREV_VSIZE	

7	6	5	4	3	2	1	0
PREV_VSIZE							

- **PREV_VSIZE: Vertical Size for the Preview Path**

PREV_VSIZE = Vertical Preview size - 1 (480 max only in RGB mode).

- **PREV_HSIZE: Horizontal Size for the Preview Path**

PREV_HSIZE = Horizontal Preview size - 1 (640 max only in RGB mode).

Downloaded from Arrow.com.

36.6.4 ISI Preview Decimation Factor Register

Name: ISI_PDECF

Address: 0x4004C00C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
DEC_FACTOR							

- **DEC_FACTOR: Decimation Factor**

DEC_FACTOR is 8-bit width, range is from 16 to 255. Values from 0 to 16 do not perform any decimation.

Downloaded from Arrow.com.

36.6.5 ISI Color Space Conversion YCrCb to RGB Set 0 Register

Name: ISI_Y2R_SET0

Address: 0x4004C010

Access: Read/Write

31	30	29	28	27	26	25	24
C3							
23	22	21	20	19	18	17	16
C2							
15	14	13	12	11	10	9	8
C1							
7	6	5	4	3	2	1	0
C0							

- **C0: Color Space Conversion Matrix Coefficient C0**

C0 element default step is 1/128, ranges from 0 to 1.9921875.

- **C1: Color Space Conversion Matrix Coefficient C1**

C1 element default step is 1/128, ranges from 0 to 1.9921875.

- **C2: Color Space Conversion Matrix Coefficient C2**

C2 element default step is 1/128, ranges from 0 to 1.9921875.

- **C3: Color Space Conversion Matrix Coefficient C3**

C3 element default step is 1/128, ranges from 0 to 1.9921875.

Downloaded from Arrow.com.

36.6.6 ISI Color Space Conversion YCrCb to RGB Set 1 Register

Name: ISI_Y2R_SET1

Address: 0x4004C014

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	Cboff	Croff	Yoff	–	–	–	C4
7	6	5	4	3	2	1	0
C4							

- **C4: Color Space Conversion Matrix Coefficient C4**

C4 element default step is 1/128, ranges from 0 to 3.9921875.

- **Yoff: Color Space Conversion Luminance Default Offset**

0: No offset.

1: Offset = 128.

- **Croff: Color Space Conversion Red Chrominance Default Offset**

0: No offset.

1: Offset = 16.

- **Cboff: Color Space Conversion Blue Chrominance Default Offset**

0: No offset.

1: Offset = 16.

Downloaded from Arrow.com.

36.6.7 ISI Color Space Conversion RGB to YCrCb Set 0 Register

Name: ISI_R2Y_SET0

Address: 0x4004C018

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	Roff

23	22	21	20	19	18	17	16
–	C2						

15	14	13	12	11	10	9	8
–	C1						

7	6	5	4	3	2	1	0
–	C0						

- **C0: Color Space Conversion Matrix Coefficient C0**

C0 element default step is 1/256, from 0 to 0.49609375.

- **C1: Color Space Conversion Matrix Coefficient C1**

C1 element default step is 1/128, from 0 to 0.9921875.

- **C2: Color Space Conversion Matrix Coefficient C2**

C2 element default step is 1/512, from 0 to 0.2480468875.

- **Roff: Color Space Conversion Red Component Offset**

0: No offset

1: Offset = 16

Downloaded from Arrow.com.

36.6.8 ISI Color Space Conversion RGB to YCrCb Set 1 Register

Name: ISI_R2Y_SET1

Address: 0x4004C01C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	Goff

23	22	21	20	19	18	17	16
–	C5						

15	14	13	12	11	10	9	8
–	C4						

7	6	5	4	3	2	1	0
–	C3						

- **C3: Color Space Conversion Matrix Coefficient C3**

C0 element default step is 1/128, ranges from 0 to 0.9921875.

- **C4: Color Space Conversion Matrix Coefficient C4**

C1 element default step is 1/256, ranges from 0 to 0.49609375.

- **C5: Color Space Conversion Matrix Coefficient C5**

C1 element default step is 1/512, ranges from 0 to 0.2480468875.

- **Goff: Color Space Conversion Green Component Offset**

0: No offset.

1: Offset = 128.

Downloaded from Arrow.com.

36.6.9 ISI Color Space Conversion RGB to YCrCb Set 2 Register

Name: ISI_R2Y_SET2

Address: 0x4004C020

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	Boff

23	22	21	20	19	18	17	16
–	C8						

15	14	13	12	11	10	9	8
–	C7						

7	6	5	4	3	2	1	0
–	C6						

- **C6: Color Space Conversion Matrix Coefficient C6**

C6 element default step is 1/512, ranges from 0 to 0.2480468875.

- **C7: Color Space Conversion Matrix Coefficient C7**

C7 element default step is 1/256, ranges from 0 to 0.49609375.

- **C8: Color Space Conversion Matrix Coefficient C8**

C8 element default step is 1/128, ranges from 0 to 0.9921875.

- **Boff: Color Space Conversion Blue Component Offset**

0: No offset.

1: Offset = 128.

Downloaded from Arrow.com.

36.6.10 ISI Control Register

Name: ISI_CR

Address: 0x4004C024

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	ISI_CDC
7	6	5	4	3	2	1	0
–	–	–	–	–	ISI_SRST	ISI_DIS	ISI_EN

- **ISI_EN: ISI Module Enable Request**

Write a one to this bit to enable the module. Software must poll the ENABLE bit in the ISI_SR to verify that the command has successfully completed.

- **ISI_DIS: ISI Module Disable Request**

Write a one to this bit to disable the module. If both ISI_EN and ISI_DIS are asserted at the same time, the disable request is not taken into account. Software must poll the DIS_DONE bit in the ISI_SR to verify that the command has successfully completed.

- **ISI_SRST: ISI Software Reset Request**

Write a one to this bit to request a software reset of the module. Software must poll the SRST bit in the ISI_SR to verify that the software request command has terminated.

- **ISI_CDC: ISI Codec Request**

Write a one to this bit to enable the codec datapath and capture a full resolution frame. A new request cannot be taken into account while CDC_PND bit is active in the ISI_SR.

Downloaded from Arrow.com.

36.6.11 ISI Status Register

Name: ISI_SR

Address: 0x4004C028

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	FR_OVR	CRC_ERR	C_OVR	P_OVR
23	22	21	20	19	18	17	16
–	–	–	–	SIP	–	CXFR_DONE	PXFR_DONE
15	14	13	12	11	10	9	8
–	–	–	–	–	VSYNC	–	CDC_PND
7	6	5	4	3	2	1	0
–	–	–	–	–	SRST	DIS_DONE	ENABLE

- **ENABLE: Module Enable**

0: Module is disabled.

1: Module is enabled.

- **DIS_DONE: Module Disable Request has Terminated (cleared on read)**

0: Indicates that the request is not completed (if a request was issued).

1: Disable request has completed. This flag is reset after a read operation.

- **SRST: Module Software Reset Request has Terminated (cleared on read)**

0: Indicates that the request is not completed (if a request was issued).

1: Software reset request has completed. This flag is reset after a read operation.

- **CDC_PND: Pending Codec Request**

0: Indicates that no codec request is pending

1: Indicates that the request has been taken into account but cannot be serviced within the current frame. The operation is postponed to the next frame.

- **VSYNC: Vertical Synchronization (cleared on read)**

0: Indicates that the vertical synchronization has not been detected since the last read of the ISI_SR.

1: Indicates that a vertical synchronization has been detected since the last read of the ISI_SR.

- **PXFR_DONE: Preview DMA Transfer has Terminated (cleared on read)**

0: Preview transfer done not detected.

1: Preview transfer done detected. When set, this bit indicates that the data transfer on the preview channel has completed since the last read of ISI_SR.

- **CXFR_DONE: Codec DMA Transfer has Terminated (cleared on read)**

0: Codec transfer done not detected.

1: Codec transfer done detected. When set, this bit indicates that the data transfer on the codec channel has completed since the last read of ISI_SR.

Downloaded from Arrow.com.

- **SIP: Synchronization in Progress**

When the status of the preview or codec DMA channel is modified, a minimum amount of time is required to perform the clock domain synchronization.

0: The clock domain synchronization process is terminated.

1: This bit is set when the clock domain synchronization operation occurs. No modification of the channel status is allowed when this bit is set, to guarantee data integrity.

- **P_OVR: Preview Datapath Overflow (cleared on read)**

0: No overflow

1: An overrun condition has occurred in input FIFO on the preview path. The overrun happens when the FIFO is full and an attempt is made to write a new sample to the FIFO since the last read of ISI_SR.

- **C_OVR: Codec Datapath Overflow (cleared on read)**

0: No overflow

1: An overrun condition has occurred in input FIFO on the codec path. The overrun happens when the FIFO is full and an attempt is made to write a new sample to the FIFO since the last read of ISI_SR.

- **CRC_ERR: CRC Synchronization Error (cleared on read)**

0: No CRC error in the embedded synchronization frame (SAV/EAV)

1: Embedded Synchronization Correction is enabled (CRC_SYNC bit is set) in the ISI_CR and an error has been detected and not corrected since the last read of ISI_SR. The frame is discarded and the ISI waits for a new one.

- **FR_OVR: Frame Rate Overrun (cleared on read)**

0: No frame overrun

1: Frame overrun. The current frame is being skipped because a vsync signal has been detected while flushing FIFOs since the last read of ISI_SR.

Downloaded from Arrow.com.

36.6.12 ISI Interrupt Enable Register

Name: ISI_IER

Address: 0x4004C02C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	FR_OVR	CRC_ERR	C_OVR	P_OVR
23	22	21	20	19	18	17	16
–	–	–	–	–	–	CXFR_DONE	PXFR_DONE
15	14	13	12	11	10	9	8
–	–	–	–	–	VSYNC	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	SRST	DIS_DONE	–

- **DIS_DONE: Disable Done Interrupt Enable**

0: No effect.

1: Enables the corresponding interrupt.

- **SRST: Software Reset Interrupt Enable**

0: No effect.

1: Enables the corresponding interrupt.

- **VSYNC: Vertical Synchronization Interrupt Enable**

0: No effect.

1: Enables the corresponding interrupt.

- **PXFR_DONE: Preview DMA Transfer Done Interrupt Enable**

0: No effect.

1: Enables the corresponding interrupt.

- **CXFR_DONE: Codec DMA Transfer Done Interrupt Enable**

0: No effect.

1: Enables the corresponding interrupt.

- **P_OVR: Preview Datapath Overflow Interrupt Enable**

0: No effect.

1: Enables the corresponding interrupt.

- **C_OVR: Codec Datapath Overflow Interrupt Enable**

0: No effect.

1: Enables the corresponding interrupt.

Downloaded from Arrow.com.

- **CRC_ERR: Embedded Synchronization CRC Error Interrupt Enable**

0: No effect.

1: Enables the corresponding interrupt.

- **FR_OVR: Frame Rate Overflow Interrupt Enable**

0: No effect.

1: Enables the corresponding interrupt.

Downloaded from Arrow.com.

36.6.13 ISI Interrupt Disable Register

Name: ISI_IDR

Address: 0x4004C030

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	FR_OVR	CRC_ERR	C_OVR	P_OVR
23	22	21	20	19	18	17	16
–	–	–	–	–	–	CXFR_DONE	PXFR_DONE
15	14	13	12	11	10	9	8
–	–	–	–	–	VSYNC	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	SRST	DIS_DONE	–

- **DIS_DONE: Disable Done Interrupt Disable**

0: No effect.

1: Disables the corresponding interrupt.

- **SRST: Software Reset Interrupt Disable**

0: No effect.

1: Disables the corresponding interrupt.

- **VSYNC: Vertical Synchronization Interrupt Disable**

0: No effect.

1: Disables the corresponding interrupt.

- **PXFR_DONE: Preview DMA Transfer Done Interrupt Disable**

0: No effect.

1: Disables the corresponding interrupt.

- **CXFR_DONE: Codec DMA Transfer Done Interrupt Disable**

0: No effect.

1: Disables the corresponding interrupt.

- **P_OVR: Preview Datapath Overflow Interrupt Disable**

0: No effect.

1: Disables the corresponding interrupt.

- **C_OVR: Codec Datapath Overflow Interrupt Disable**

0: No effect.

1: Disables the corresponding interrupt.

Downloaded from Arrow.com.

- **CRC_ERR: Embedded Synchronization CRC Error Interrupt Disable**

0: No effect.

1: Disables the corresponding interrupt.

- **FR_OVR: Frame Rate Overflow Interrupt Disable**

0: No effect.

1: Disables the corresponding interrupt.

Downloaded from Arrow.com.

36.6.14 ISI Interrupt Mask Register

Name: ISI_IMR

Address: 0x4004C034

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	FR_OVR	CRC_ERR	C_OVR	P_OVR
23	22	21	20	19	18	17	16
–	–	–	–	–	–	CXFR_DONE	PXFR_DONE
15	14	13	12	11	10	9	8
–	–	–	–	–	VSYNC	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	SRST	DIS_DONE	–

- **DIS_DONE: Module Disable Operation Completed**

0: The Module Disable Operation Completed interrupt is disabled.

1: The Module Disable Operation Completed interrupt is enabled.

- **SRST: Software Reset Completed**

0: The Software Reset Completed interrupt is disabled.

1: The Software Reset Completed interrupt is enabled.

- **VSYNC: Vertical Synchronization**

0: The Vertical Synchronization interrupt is disabled.

1: The Vertical Synchronization interrupt is enabled.

- **PXFR_DONE: Preview DMA Transfer Completed**

0: The Preview DMA Transfer Completed interrupt is disabled.

1: The Preview DMA Transfer Completed interrupt is enabled.

- **CXFR_DONE: Codec DMA Transfer Completed**

0: The Codec DMA Transfer Completed interrupt is disabled.

1: The Codec DMA Transfer Completed interrupt is enabled.

- **P_OVR: Preview FIFO Overflow**

0: The Preview FIFO Overflow interrupt is disabled.

1: The Preview FIFO Overflow interrupt is enabled.

- **C_OVR: Codec FIFO Overflow**

0: The Codec FIFO Overflow interrupt is disabled.

1: The Codec FIFO Overflow interrupt is enabled.

Downloaded from Arrow.com.

- **CRC_ERR: CRC Synchronization Error**

0: The CRC Synchronization Error interrupt is disabled.

1: The CRC Synchronization Error interrupt is enabled.

- **FR_OVR: Frame Rate Overrun**

0: The Frame Rate Overrun interrupt is disabled.

1: The Frame Rate Overrun is enabled.

Downloaded from Arrow.com.

36.6.15 DMA Channel Enable Register

Name: ISI_DMA_CHER

Address: 0x4004C038

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	C_CH_EN	P_CH_EN

- **P_CH_EN: Preview Channel Enable**

Write a one to this bit to enable the preview DMA channel.

- **C_CH_EN: Codec Channel Enable**

Write a one to this bit to enable the codec DMA channel.

Downloaded from Arrow.com.

36.6.16 DMA Channel Disable Register

Name: ISI_DMA_CHDR

Address: 0x4004C03C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	C_CH_DIS	P_CH_DIS

- **P_CH_DIS: Preview Channel Disable Request**

0: No effect.

1: Disables the channel. Poll P_CH_S in DMA_CHSR to verify that the preview channel status has been successfully modified.

- **C_CH_DIS: Codec Channel Disable Request**

0: No effect.

1: Disables the channel. Poll C_CH_S in DMA_CHSR to verify that the codec channel status has been successfully modified.

Downloaded from Arrow.com.

36.6.17 DMA Channel Status Register

Name: ISI_DMA_CHSR

Address: 0x4004C040

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	C_CH_S	P_CH_S

- **P_CH_S: Preview DMA Channel Status**

0: Indicates that the Preview DMA channel is disabled.

1: Indicates that the Preview DMA channel is enabled.

- **C_CH_S: Code DMA Channel Status**

0: Indicates that the Codec DMA channel is disabled.

1: Indicates that the Codec DMA channel is enabled.

Downloaded from Arrow.com.

36.6.18 DMA Preview Base Address Register

Name: ISI_DMA_P_ADDR

Address: 0x4004C044

Access: Read/Write

31	30	29	28	27	26	25	24
P_ADDR							

23	22	21	20	19	18	17	16
P_ADDR							

15	14	13	12	11	10	9	8
P_ADDR							

7	6	5	4	3	2	1	0
P_ADDR						–	–

- **P_ADDR: Preview Image Base Address**

This address is word-aligned.

Downloaded from Arrow.com.

36.6.19 DMA Preview Control Register

Name: ISI_DMA_P_CTRL

Address: 0x4004C048

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	P_DONE	P_IEN	P_WB	P_FETCH

- **P_FETCH: Descriptor Fetch Control Bit**

0: Preview channel fetch operation is disabled.

1: Preview channel fetch operation is enabled.

- **P_WB: Descriptor Writeback Control Bit**

0: Preview channel writeback operation is disabled.

1: Preview channel writeback operation is enabled.

- **P_IEN: Transfer Done Flag Control**

0: Preview transfer done flag generation is enabled.

1: Preview transfer done flag generation is disabled.

- **P_DONE: Preview Transfer Done**

This bit is only updated in the memory.

0: The transfer related to this descriptor has not been performed.

1: The transfer related to this descriptor has completed. This bit is updated in memory at the end of the transfer, when writeback operation is enabled.

Downloaded from Arrow.com.

36.6.20 DMA Preview Descriptor Address Register

Name: ISI_DMA_P_DSCR

Address: 0x4004C04C

Access: Read/Write

31	30	29	28	27	26	25	24
P_DSCR							

23	22	21	20	19	18	17	16
P_DSCR							

15	14	13	12	11	10	9	8
P_DSCR							

7	6	5	4	3	2	1	0
P_DSCR						–	–

- **P_DSCR: Preview Descriptor Base Address**

This address is word-aligned.

Downloaded from Arrow.com.

36.6.21 DMA Codec Base Address Register

Name: ISI_DMA_C_ADDR

Address: 0x4004C050

Access: Read/Write

31	30	29	28	27	26	25	24
C_ADDR							

23	22	21	20	19	18	17	16
C_ADDR							

15	14	13	12	11	10	9	8
C_ADDR							

7	6	5	4	3	2	1	0
C_ADDR						–	–

- **C_ADDR: Codec Image Base Address**

This address is word-aligned.

Downloaded from Arrow.com.

36.6.22 DMA Codec Control Register

Name: ISI_DMA_C_CTRL

Address: 0x4004C054

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	C_DONE	C_IEN	C_WB	C_FETCH

- **C_FETCH: Descriptor Fetch Control Bit**

0: Codec channel fetch operation is disabled.

1: Codec channel fetch operation is enabled.

- **C_WB: Descriptor Writeback Control Bit**

0: Codec channel writeback operation is disabled.

1: Codec channel writeback operation is enabled.

- **C_IEN: Transfer Done Flag Control**

0: Codec transfer done flag generation is enabled.

1: Codec transfer done flag generation is disabled.

- **C_DONE: Codec Transfer Done**

This bit is only updated in the memory.

0: The transfer related to this descriptor has not been performed.

1: The transfer related to this descriptor has completed. This bit is updated in memory at the end of the transfer when write-back operation is enabled.

Downloaded from Arrow.com.

36.6.23 DMA Codec Descriptor Address Register

Name: ISI_DMA_C_DSCR

Address: 0x4004C058

Access: Read/Write

31	30	29	28	27	26	25	24
C_DSCR							

23	22	21	20	19	18	17	16
C_DSCR							

15	14	13	12	11	10	9	8
C_DSCR							

7	6	5	4	3	2	1	0
C_DSCR						–	–

- **C_DSCR: Codec Descriptor Base Address**

This address is word-aligned.

Downloaded from Arrow.com.

36.6.24 ISI Write Protection Mode Register

Name: ISI_WPMR

Address: 0x4004C0E4

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							
23	22	21	20	19	18	17	16
WPKEY							
15	14	13	12	11	10	9	8
WPKEY							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x495349 (“ISI” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x495349 (“ISI” in ASCII).

- **WPKEY: Write Protection Key Password**

Value	Name	Description
0x495349	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

Downloaded from Arrow.com.

36.6.25 ISI Write Protection Status Register

Name: ISI_WPSR

Address: 0x4004C0E8

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
WPVSRC							
15	14	13	12	11	10	9	8
WPVSRC							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

- **WPVS: Write Protection Violation Status**

Value	Description
0	No write protection violation occurred since the last read of ISI_WPSR.
1	A write protection violation has occurred since the last read of the ISI_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protection Violation Source**

Value	Description
0	No Write Protection Violation occurred since the last read of this register (ISI_WPSR).
1	Write access in ISI_CFG1 while Write Protection was enabled (since the last read).
2	Write access in ISI_CFG2 while Write Protection was enabled (since the last read).
3	Write access in ISI_PSIZE while Write Protection was enabled (since the last read).
4	Write access in ISI_PDECF while Write Protection was enabled (since the last read).
5	Write access in ISI_Y2R_SET0 while Write Protection was enabled (since the last read).
6	Write access in ISI_Y2R_SET1 while Write Protection was enabled (since the last read).
7	Write access in ISI_R2Y_SET0 while Write Protection was enabled (since the last read).
8	Write access in ISI_R2Y_SET1 while Write Protection was enabled (since the last read).
9	Write access in ISI_R2Y_SET2 while Write Protection was enabled (since the last read).

Downloaded from Arrow.com.

37. USB High-Speed Interface (USBHS)

37.1 Description

The USB High-Speed Interface (USBHS) complies with the Universal Serial Bus (USB) 2.0 specification in all speeds.

Each pipe/endpoint can be configured in one of several USB transfer types. It can be associated with one, two or three banks of a DPRAM used to store the current data payload. If two or three banks are used, then one DPRAM bank is read or written by the CPU or the DMA, while the other is read or written by the USBHS core. This feature is mandatory for isochronous pipes/endpoints.

Table 37-1 describes the hardware configuration of the USB MCU device.

Table 37-1. Description of USB Pipes/Endpoints

Pipe/Endpoint	Mnemonic	Max. Nb. Banks	DMA	High Band Width	Max. Pipe/ Endpoint Size	Type
0	PEP_0	1	N	N	64	Control
1	PEP_1	3	Y	Y	1024	Isochronous/Bulk/Interrupt/Control
2	PEP_2	3	Y	Y	1024	Isochronous/Bulk/Interrupt/Control
3	PEP_3	2	Y	Y	1024	Isochronous/Bulk/Interrupt/Control
4	PEP_4	2	Y	Y	1024	Isochronous/Bulk/Interrupt/Control
5	PEP_5	2	Y	Y	1024	Isochronous/Bulk/Interrupt/Control
6	PEP_6	2	Y	Y	1024	Isochronous/Bulk/Interrupt/Control
7	PEP_7	2	Y	Y	1024	Isochronous/Bulk/Interrupt/Control
8	PEP_8	2	N	Y	1024	Isochronous/Bulk/Interrupt/Control
9	PEP_9	2	N	Y	1024	Isochronous/Bulk/Interrupt/Control

37.2 Embedded Characteristics

- Compatible with the USB 2.0 Specification
- Supports High-Speed (480 Mbps), Full-Speed (12 Mbps) and Low-Speed (1.5 Mbps) Communication
- 10 Pipes/Endpoints
- 4096 bytes of Embedded Dual-Port RAM (DPRAM) for Pipes/Endpoints
- Up to 3 Memory Banks per Pipe/Endpoint (not for Control Pipe/Endpoint)
- Flexible Pipe/Endpoint Configuration and Management with Dedicated DMA Channels
- On-Chip UTMI Transceiver including Pull-ups/Pull-downs

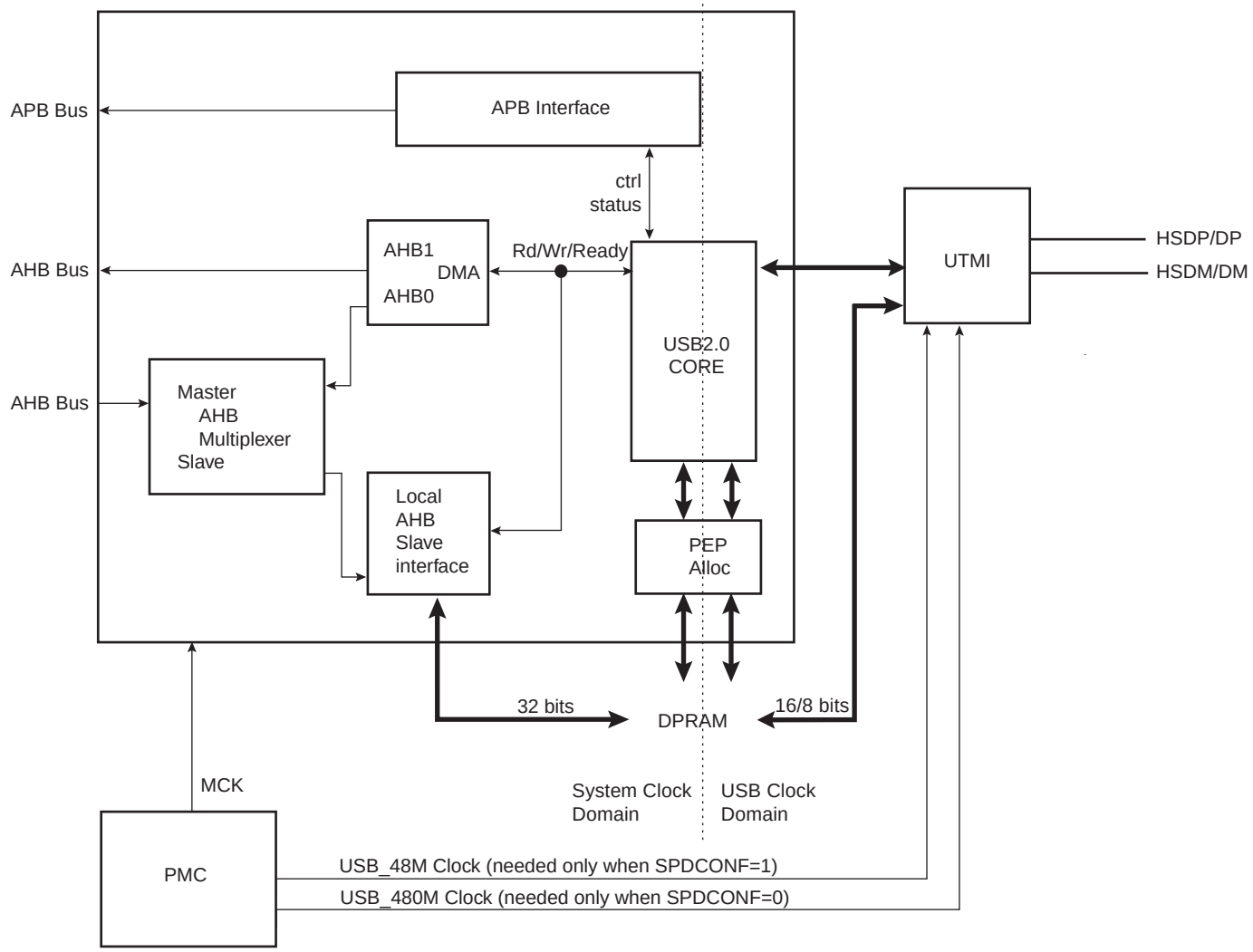
Downloaded from Arrow.com.

37.3 Block Diagram

The USBHS provides a hardware device to interface a USB link to a data flow stored in a dual-port RAM (DPRAM).

In normal operation (SPDCONF = 0), the UTMI transceiver requires the UTMI PLL (480 MHz). In case of full-speed or low-speed only, for a lower consumption (SPDCONF = 1), the UTMI transceiver only requires 48 MHz.

Figure 37-1. USBHS Block Diagram



37.3.1 Signal Description

Table 37-2. Signal Description

Name	Description	Type
HSDM/DM	HS/FS Differential Data Line -	Input/Output
HSDP/DP	HS/FS Differential Data Line +	Input/Output

Downloaded from Arrow.com.

37.4 Product Dependencies

37.4.1 I/O Lines

A regular PIO line must be used to control VBUS. This is configured in the I/O Controller.

37.4.2 Clocks

The clock for the USBHS bus interface is generated by the Power Management Controller. This clock can be enabled or disabled in the Power Management Controller. It is recommended to disable the USBHS before disabling the clock, to avoid freezing the USBHS in an undefined state.

Before enabling the USB clock in the Power Management Controller, the USBHS must be enabled (by writing a one to the USBHS_CTRL.USBE bit and a zero to the USBHS_CTRL.FRZCLK bit).

The USBHS can work in two modes:

- Normal mode (SPDCONF = 0) where High speed, Full speed and Low speed are available.
- Low-power mode (SPDCONF = 1) where Full speed and Low speed are available.

To ensure successful start-up, follow the sequences below:

- In Normal mode:

1. Enable the USBHS peripheral clock. This is done via the register PMC_PCER.
2. Enable the USBHS (UIMOD, USBE = 1, FRZCLK = 0).
3. Enable the UPLL 480 MHz.
4. Wait for the UPLL 480 MHz to be considered as locked by the PMC.

- In Low-power mode:

1. As USB_48M must be set to 48 MHz (refer to Section 31. "Power Management Controller (PMC)"), select either the PLLA or the UPLL (previously set to ON), and program the PMC_USB register (source selection and divider).
2. Enable the USBHS peripheral clock (PMC_PCER).
3. Put the USBHS in Low-power mode (SPDCONF = 1).
4. Enable the USBHS (UIMOD, USBE = 1, FRZCLK = 0).
5. Enable the USBCK bit (PMC_SCER).

37.4.3 Interrupt Sources

The USBHS interrupt request line is connected to the interrupt controller. Using the USBHS interrupt requires the interrupt controller to be programmed first.

Table 37-3. Peripheral IDs

Instance	ID
USBHS	34

37.4.4 USB Pipe/Endpoint x FIFO Data Register (USBFIFOxDATA)

The application has access to each pipe/endpoint FIFO through its reserved 32 KB address space. The application can access a 64-KB buffer linearly or fixedly as the DPRAM address increment is fully handled by hardware. Byte, half-word and word accesses are supported. Data should be accessed in a big-endian way.

Disabling the USBHS (by writing a zero to the USBHS_CTRL.USBE bit) does not reset the DPRAM.

Downloaded from Arrow.com.

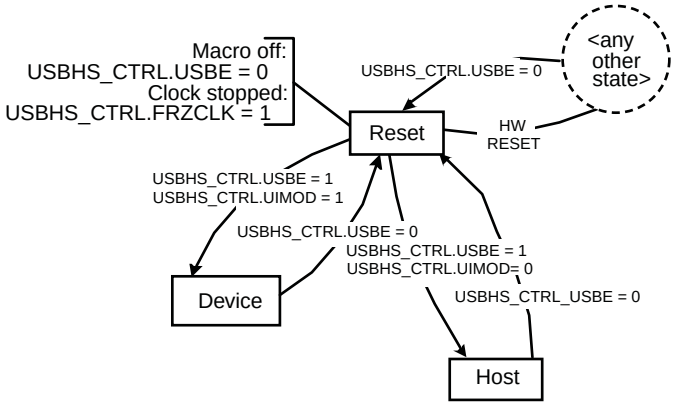
37.5 Functional Description

37.5.1 USB General Operation

37.5.1.1 Power-On and Reset

Figure 37-2 describes the USBHS general states.

Figure 37-2. General States



After a hardware reset, the USBHS is in Reset state. In this state:

- The USBHS is disabled. The USBHS Enable bit in the General Control register (USBHS_CTRL.USBE) is zero.
- The USBHS clock is stopped in order to minimize power consumption. The Freeze USB Clock bit (USBHS_CTRL.FRZCLK) is set.
- The UTMI is in Suspend mode.
- The internal states and registers of the Device and Host modes are reset.
- The DPRAM is not cleared and is accessible.

After writing a one to USBHS_CTRL.USBE, the USBHS enters the Device or the Host mode in idle state.

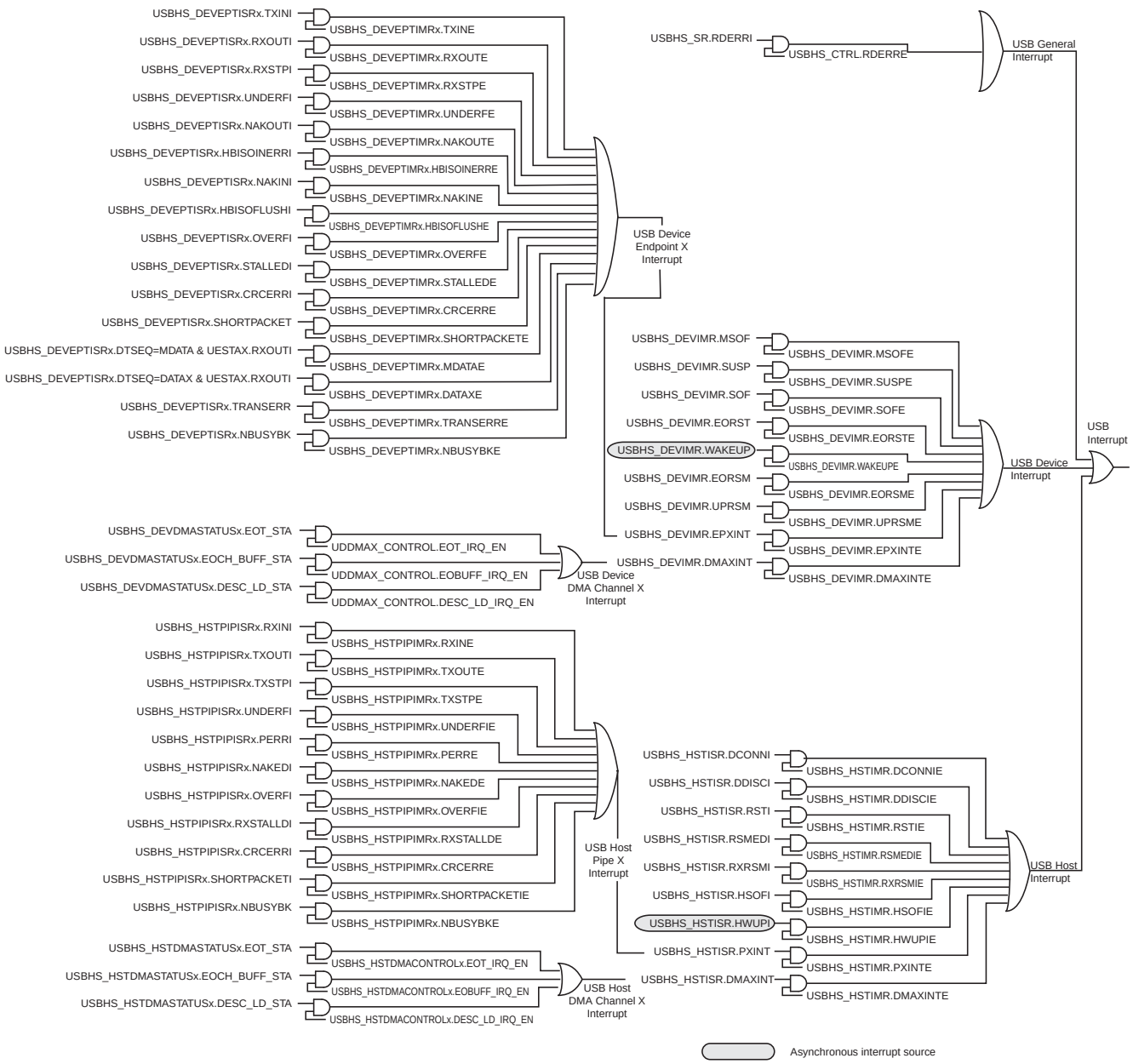
The USBHS can be disabled at any time by writing a zero to USBHS_CTRL.USBE. This acts as a hardware reset, except that the USBHS_CTRL.FRZCLK, USBHS_CTRL.UIMOD and USBHS_DEVCTRL.LS bits are not reset.

Downloaded from Arrow.com.

37.5.1.2 Interrupts

One interrupt vector is assigned to the USB interface. Figure 37-3 shows the structure of the USB interrupt system.

Figure 37-3. Interrupt System



See Section 37.5.2.19 and Section 37.5.3.13 for further details about device and host interrupts.

There are two kinds of general interrupts: processing, i.e., their generation is part of the normal processing, and exception, i.e., errors (not related to CPU exceptions).

37.5.1.3 MCU Power Modes

USB Suspend Mode

In Peripheral mode, the Suspend Interrupt bit in the Device Global Interrupt Status register (USBHS_DEVISR.SUSP) indicates that the USB line is in Suspend mode. In this case, the transceiver is automatically set in Suspend mode to reduce consumption.

Downloaded from Arrow.com.

Clock Frozen

The USBHS can be frozen when the USB line is in the Suspend mode, by writing a one to the USBHS_CTRL.FRZCLK bit, which reduces power consumption.

In this case, it is still possible to access the following:

- USBHS_CTRL.FRZCLK, USBHS_CTRL.USBE and USBHS_DEVCTRL.LS bits

Moreover, when USBHS_CTRL.FRZCLK = 1, only the asynchronous interrupt sources can trigger the USB interrupt:

- Wake-up Interrupt (USBHS_DEVISR.WAKEUP)
- Host Wake-up Interrupt (USBHS_HSTISR.HWUPI)

37.5.1.4 Speed Control

Device Mode

When the USB interface is in Device mode, the speed selection (Full-speed or High-speed) is performed automatically by the USBHS during the USB reset according to the host speed capability. At the end of the USB reset, the USBHS enables or disables high-speed terminations and pull-up.

It is possible to set the USBHS to Full-speed or Low-speed mode via USBHS_DEVCTRL.LS and USBHS_DEVCTRL.SPDCONF.

Host Mode

When the USB interface is in Host mode, internal pull-down resistors are connected on both D+ and D- and the interface detects the speed of the connected device, which is reflected by the Speed Status (USBHS_SR.SPEED) field.

37.5.1.5 DPRAM Management

Pipes and endpoints can only be allocated in ascending order, from pipe/endpoint 0 to the last pipe/endpoint to be allocated. The user should therefore configure them in the same order.

The allocation of a pipe/endpoint x starts when the Endpoint Memory Allocate bit in the Endpoint x Configuration register (USBHS_DEVEPTCFGx.ALLOC) is written to one. Then, the hardware allocates a memory area in the DPRAM and inserts it between the x-1 and x+1 pipes/endpoints. The x+1 pipe/endpoint memory window slides up and its data is lost. Note that the following pipe/endpoint memory windows (from x+2) do not slide.

Disabling a pipe, by writing a zero to the Pipe x Enable bit in the Host Pipe register (USBHS_HSTPIP.PENx), or disabling an endpoint, by writing a zero to the Endpoint x Enable bit in the Device Endpoint register (USBHS_DEVEPT.EPENx), does not reset the USBHS_DEVEPTCFGx.ALLOC bit or the Pipe/Endpoint configuration:

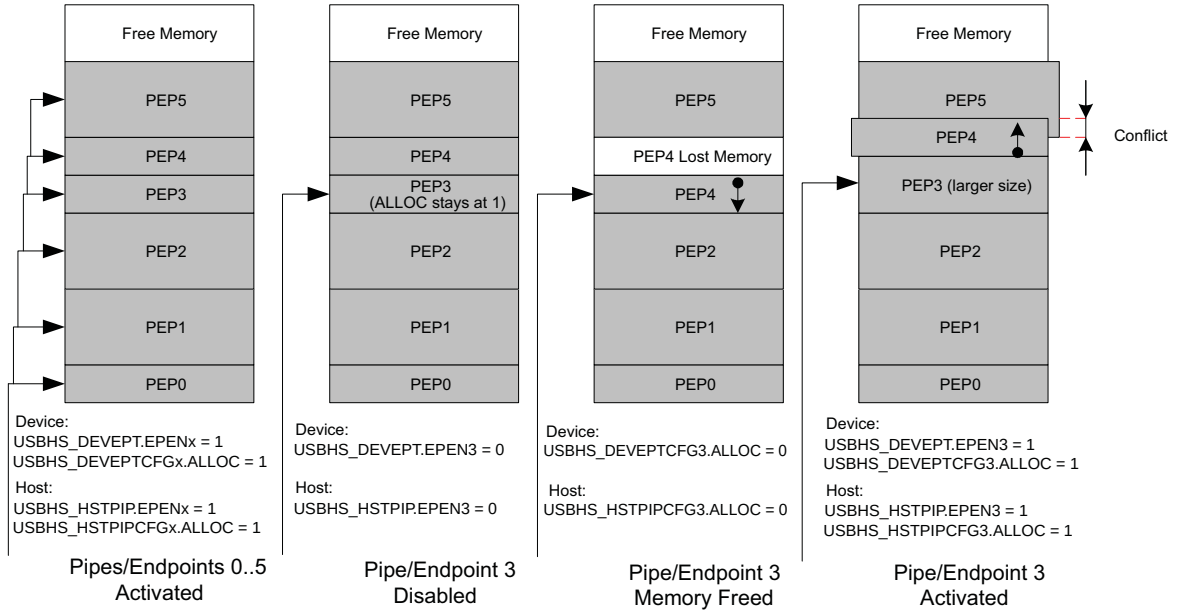
- Pipe Configuration
  - Pipe Banks (USBHS_HSTPIPCFGx.PBK)
  - Pipe Size (USBHS_HSTPIPCFGx.PSIZE)
  - Pipe Token (USBHS_HSTPIPCFGx.PTOKEN)
  - Pipe Type (USBHS_HSTPIPCFGx.PTYPE)
  - Pipe Endpoint Number (USBHS_HSTPIPCFGx.PEPNUM)
  - Pipe Interrupt Request Frequency (USBHS_HSTPIPCFGx.INTFRQ)
- Endpoint Configuration
  - Endpoint Banks (USBHS_DEVEPTCFGx.EPBK)
  - Endpoint Size (USBHS_DEVEPTCFGx. EPSIZE)
  - Endpoint Direction (USBHS_DEVEPTCFGx.EPDIR)
  - Endpoint Type (USBHS_DEVEPTCFGx.EPTYPE)

Downloaded from Arrow.com.

To free endpoint memory, the user must write a zero to the USBHS_DEVEPTCFGx.ALLOC bit. The x+1 pipe/endpoint memory window then slides down and its data is lost. Note that the following pipe/endpoint memory windows (from x+2) do not slide.

Figure 37-4 illustrates the allocation and reorganization of the DPRAM in a typical example.

Figure 37-4. Allocation and Reorganization of the DPRAM



1. Pipes/endpoints 0 to 5 are enabled, configured and allocated in ascending order. Each pipe/endpoint then owns a memory area in the DPRAM.
2. Pipe/endpoint 3 is disabled, but its memory is kept allocated by the controller.
3. In order to free its memory, its USBHS_DEVEPTCFGx.ALLOC bit is written to zero. The pipe/endpoint 4 memory window slides down, but pipe/endpoint 5 does not move.
4. If the user chooses to reconfigure pipe/endpoint 3 with a larger size, the controller allocates a memory area after the pipe/endpoint 2 memory area and automatically slides up the pipe/endpoint 4 memory window. Pipe/endpoint 5 does not move and a memory conflict appears as the memory windows of pipes/endpoints 4 and 5 overlap. The data of these pipes/endpoints is potentially lost.

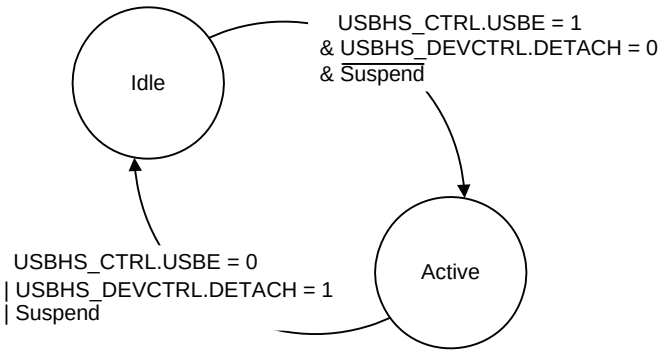
Note:
1. The data of pipe/endpoint 0 cannot be lost (except if it is de-allocated) as the memory allocation and de-allocation may affect only higher pipes/endpoints.
2. Deactivating then reactivating the same pipe/endpoint with the same configuration only modifies temporarily the controller DPRAM pointer and size for this pipe/endpoint. Nothing changes in the DPRAM. Higher endpoints seem not to have been moved and their data is preserved as long as nothing has been written or received into them while changing the allocation state of the first pipe/endpoint.
3. When the user writes a one to the USBHS_DEVEPTCFGx.ALLOC bit, the Configuration OK Status bit (USBHS_DEVEPTISRx.CFGOK) is set only if the configured size and number of banks are correct as compared to the endpoint maximum allowed values and to the maximum FIFO size (i.e., the DPRAM size). The USBHS_DEVEPTISRx.CFGOK value does not consider memory allocation conflicts.

Downloaded from Arrow.com.

37.5.1.6 Pad Suspend

Figure 37-5 shows the pad behavior.

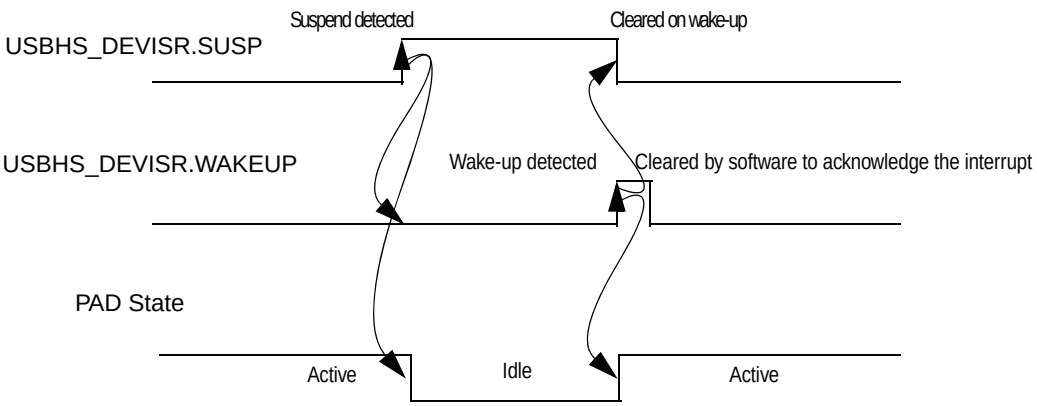
Figure 37-5. Pad Behavior



- In Idle state, the pad is put in Low-power mode, i.e., the differential receiver of the USB pad is off, and internal pull-downs with a strong value (15 K) are set in HSDP/D and HSDM/DM to avoid floating lines.
- In Active state, the pad is working.

Figure 37-6 illustrates the pad events leading to a PAD state change.

Figure 37-6. Pad Events



The USBHS_DEVISR.SUSP bit is set and the Wake-Up Interrupt (USBHS_DEVISR.WAKEUP) bit is cleared when a USB “Suspend” state has been detected on the USB bus. This event automatically puts the USB pad in Idle state. The detection of a non-idle event sets USBHS_DEVISR.WAKEUP, clears USBHS_DEVISR.SUSP and wakes up the USB pad.

The pad goes to the Idle state if the USBHS is disabled or if the USBHS_DEVCTRL.DETACH bit = 1. It returns to the Active state when USBHS_CTRL.USBE = 1 and USBHS_DEVCTRL.DETACH = 0.

Downloaded from Arrow.com.

37.5.2 USB Device Operation

37.5.2.1 Introduction

In Device mode, the USBHS supports high-, full- and low-speed data transfers.

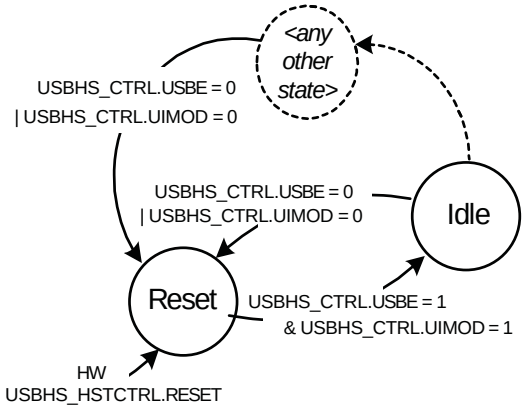
In addition to the default control endpoint, 10 endpoints are provided, which can be configured with an isochronous, bulk or interrupt type, as described in Table 37-1 on page 570.

As the Device mode starts in Idle state, the pad consumption is reduced to the minimum.

37.5.2.2 Power-On and Reset

Figure 37-7 describes the USBHS Device mode main states.

Figure 37-7. Device Mode Main States



After a hardware reset, the USBHS Device mode is in Reset state. In this state:

- the USBHS clock is stopped to minimize power consumption (USBHS_CTRL.FRZCLK = 1),
- the internal registers of the Device mode are reset,
- the endpoint banks are de-allocated,
- neither D+ nor D- is pulled up (USBHS_DEVCTRL.DETACH = 1).

D+ or D- is pulled up according to the selected speed as soon as the USBHS_DEVCTRL.DETACH bit is written to zero. See “Device Mode” for further details.

When the USBHS is enabled (USBHS_CTRL.USBE = 1) in Device mode (USBHS_CTRL.UIMOD = 1), its Device mode state enters Idle state with minimal power consumption. This does not require the USB clock to be activated.

The USBHS Device mode can be disabled and reset at any time by disabling the USBHS (by writing a zero to USBHS_CTRL.USBE) or when the Host mode is enabled (USBHS_CTRL.UIMOD = 0).

37.5.2.3 USB Reset

The USB bus reset is managed by hardware. It is initiated by a connected host.

When a USB reset is detected on the USB line, the following operations are performed by the controller:

- All endpoints are disabled, except the default control endpoint.
- The default control endpoint is reset (see Section 37.5.2.4 for more details).
- The data toggle sequence of the default control endpoint is cleared.
- At the end of the reset process, the End of Reset (USBHS_DEVISR.EORST) bit is set.

Downloaded from Arrow.com.

- During a reset, the USBHS automatically switches to High-speed mode if the host is High-speed-capable (the reset is called High-speed reset). The user should observe the USBHS_SR.SPEED field to know the speed running at the end of the reset (USBHS_DEVISR.EORST = 1).

37.5.2.4 Endpoint Reset

An endpoint can be reset at any time by writing a one to the Endpoint x Reset bit USBHS_DEVEPT.EPRSTx. This is recommended before using an endpoint upon hardware reset or when a USB bus reset has been received. This resets:

- the internal state machine of the endpoint,
- the receive and transmit bank FIFO counters,
- all registers of this endpoint (USBHS_DEVEPTCFGx, USBHS_DEVEPTISRx, the Endpoint x Control (USBHS_DEVEPTIMRx) register), except its configuration (USBHS_DEVEPTCFGx.ALLOC, USBHS_DEVEPTCFGx.EPBK, USBHS_DEVEPTCFGx.EPSIZE, USBHS_DEVEPTCFGx.EPDIR, USBHS_DEVEPTCFGx.EPTYPE) and the Data Toggle Sequence (USBHS_DEVEPTISRx.DTSEQ) field.

Note: The interrupt sources located in USBHS_DEVEPTISRx are not cleared when a USB bus reset has been received.

The endpoint configuration remains active and the endpoint is still enabled.

The endpoint reset may be associated with a clear of the data toggle sequence as an answer to the CLEAR_FEATURE USB request. This can be achieved by writing a one to the Reset Data Toggle Set bit (RSTDTS) in the Device Endpoint x Control Set register (this sets the Reset Data Toggle bit USBHS_DEVEPTIMRx.RSTDT).

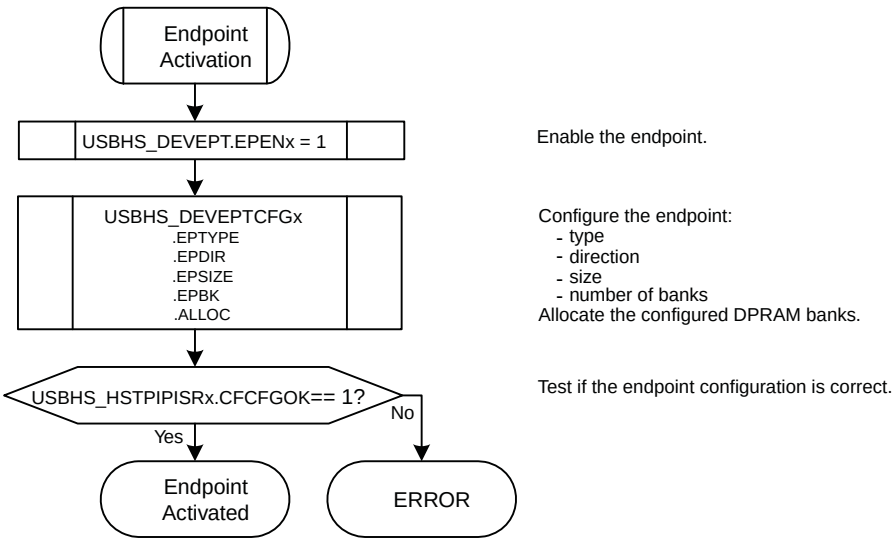
In the end, the user has to write a zero to the USBHS_DEVEPT.EPRSTx bit to complete the reset operation and to start using the FIFO.

37.5.2.5 Endpoint Activation

The endpoint is maintained inactive and reset (see Section 37.5.2.4 for more details) as long as it is disabled (USBHS_DEVEPT.EPENx = 0). USBHS_DEVEPTISRx.DTSEQ is also reset.

The algorithm represented on Figure 37-8 must be followed in order to activate an endpoint.

Figure 37-8. Endpoint Activation Algorithm



As long as the endpoint is not correctly configured (USBHS_HSTPIPISRx.CFGOK = 0), the controller does not acknowledge the packets sent by the host to this endpoint.

Downloaded from Arrow.com.

The USBHS_HSTPIPISRx.CFGOK bit is set provided that the configured size and number of banks are correct as compared to the endpoint maximal allowed values (see Table 37-1 on page 570) and to the maximal FIFO size (i.e., the DPRAM size).

See Section 37.5.1.5 for more details about DPRAM management.

37.5.2.6 Address Setup

The USB device address is set up according to the USB protocol.

- After all kinds of resets, the USB device address is 0.
- The host starts a SETUP transaction with a SET_ADDRESS (addr) request.
- The user writes this address to the USB Address (USBHS_DEVCTRL.UADD) field, and writes a zero to the Address Enable (USBHS_DEVCTRL.ADDEN) bit, so the actual address is still 0.
- The user sends a zero-length IN packet from the control endpoint.
- The user enables the recorded USB device address by writing a one to USBHS_DEVCTRL.ADDEN.

Once the USB device address is configured, the controller filters the packets to accept only those targeting the address stored in USBHS_DEVCTRL.UADD.

USBHS_DEVCTRL.UADD and USBHS_DEVCTRL.ADDEN must not be written all at once.

USBHS_DEVCTRL.UADD and USBHS_DEVCTRL.ADDEN are cleared:

- on a hardware reset,
- when the USBHS is disabled (USBHS_CTRL.USBE = 0),
- when a USB reset is detected.

When USBHS_DEVCTRL.UADD or USBHS_DEVCTRL.ADDEN is cleared, the default device address 0 is used.

37.5.2.7 Suspend and Wake-up

When an idle USB bus state has been detected for 3 ms, the controller sets the Suspend (USBHS_DEVISR.SUSP) interrupt bit. The user may then write a one to the USBHS_CTRL.FRZCLK bit to reduce power consumption.

To recover from the Suspend mode, the user should wait for the Wake-Up (USBHS_DEVISR.WAKEUP) interrupt bit, which is set when a non-idle event is detected, then write a zero to USBHS_CTRL.FRZCLK.

As the USBHS_DEVISR.WAKEUP interrupt bit is set when a non-idle event is detected, it can occur whether the controller is in the Suspend mode or not. The USBHS_DEVISR.SUSP and USBHS_DEVISR.WAKEUP interrupts are thus independent, except that one bit is cleared when the other is set.

37.5.2.8 Detach

The reset value of the USBHS_DEVCTRL.DETACH bit is one.

It is possible to initiate a device re-enumeration by simply writing a one, and then a zero, to USBHS_DEVCTRL.DETACH.

USBHS_DEVCTRL.DETACH acts on the pull-up connections of the D+ and D- pads. See “Device Mode” for further details.

37.5.2.9 Remote Wake-up

The Remote Wake-Up request (also known as Upstream Resume) is the only one the device may send without a host invitation, assuming a host command allowing the device to send such a request was previously issued. The sequence is the following:

Downloaded from Arrow.com.

1. The USBHS must have detected a “Suspend” state on the bus, i.e., the Remote Wake-Up request can only be sent after a USBHS_DEVISR.SUSP interrupt has been set.
2. The user writes a one to the Remote Wake-Up (USBHS_DEVCTRL.RMWKUP) bit to send an upstream resume to the host for a remote wake-up. This will automatically be done by the controller after 5ms of inactivity on the USB bus.
3. When the controller sends the upstream resume, the Upstream Resume (USBHS_DEVISR.UPRSM) interrupt is set and USBHS_DEVISR.SUSP is cleared.
4. USBHS_DEVCTRL.RMWKUP is cleared at the end of the upstream resume.
5. When the controller detects a valid “End of Resume” signal from the host, the End of Resume (USBHS_DEVISR.EORSM) interrupt is set.

37.5.2.10 STALL Request

For each endpoint, the STALL management is performed using:

- the STALL Request (USBHS_DEVEPTIMRx.STALLRQ) bit to initiate a STALL request,
- the STALLed Interrupt (USBHS_DEVEPTISRx.STALLEDI) bit, which is set when a STALL handshake has been sent.

To answer the next request with a STALL handshake, USBHS_DEVEPTIMRx.STALLRQ has to be set by writing a one to the STALL Request Set (USBHS_DEVEPTIERx.STALLRQS) bit. All following requests are discarded (USBHS_DEVEPTISRx.RXOUTI, etc. is not be set) and handshaked with a STALL until the USBHS_DEVEPTIMRx.STALLRQ bit is cleared, which is done when a new SETUP packet is received (for control endpoints) or when the STALL Request Clear (USBHS_DEVEPTIMRx.STALLRQC) bit is written to one.

Each time a STALL handshake is sent, the USBHS_DEVEPTISRx.STALLEDI bit is set by the USBHS and the PEP_x interrupt is set.

Special Considerations for Control Endpoints

If a SETUP packet is received into a control endpoint for which a STALL is requested, the Received SETUP Interrupt (USBHS_DEVEPTISRx.RXSTPI) bit is set and USBHS_DEVEPTIMRx.STALLRQ and USBHS_DEVEPTISRx.STALLEDI are cleared. The SETUP has to be ACKed.

This simplifies the enumeration process management. If a command is not supported or contains an error, the user requests a STALL and can return to the main task, waiting for the next SETUP request.

STALL Handshake and Retry Mechanism

The retry mechanism has priority over the STALL handshake. A STALL handshake is sent if the USBHS_DEVEPTIMRx.STALLRQ bit is set and if no retry is required.

37.5.2.11 Management of Control Endpoints

Overview

A SETUP request is always ACKed. When a new SETUP packet is received, the USBHS_DEVEPTISRx.RXSTPI is set; the Received OUT Data Interrupt (USBHS_DEVEPTISRx.RXOUTI) bit is not.

The FIFO Control (USBHS_DEVEPTIMRx.FIFOCON) bit and the Read/Write Allowed (USBHS_DEVEPTISRx.RWALL) bit are irrelevant for control endpoints. The user never uses them on these endpoints. When read, their values are always zero.

Control endpoints are managed using:

- the USBHS_DEVEPTISRx.RXSTPI bit, which is set when a new SETUP packet is received and which is cleared by firmware to acknowledge the packet and to free the bank;
- the USBHS_DEVEPTISRx.RXOUTI bit, which is set when a new OUT packet is received and which is cleared by firmware to acknowledge the packet and to free the bank;

Downloaded from Arrow.com.

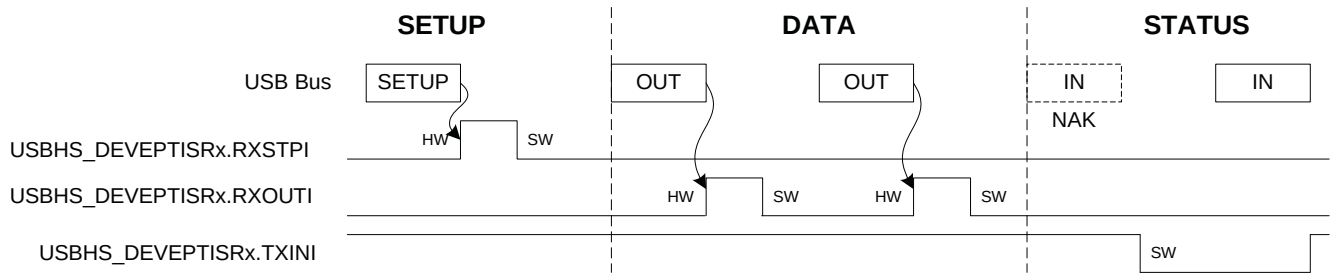
- the Transmitted IN Data Interrupt (USBHS_DEVEPTISRx.TXINI) bit, which is set when the current bank is ready to accept a new IN packet and which is cleared by firmware to send the packet.

Control Write

Figure 37-9 shows a control write transaction. During the status stage, the controller does not necessarily send a NAK on the first IN token:

- if the user knows the exact number of descriptor bytes that must be read, it can then anticipate the status stage and send a zero-length packet after the next IN token, or
- it can read the bytes and wait for the NAKed IN Interrupt (USBHS_DEVEPTISRx.NAKINI), which acknowledges that all the bytes have been sent by the host and that the transaction is now in the status stage.

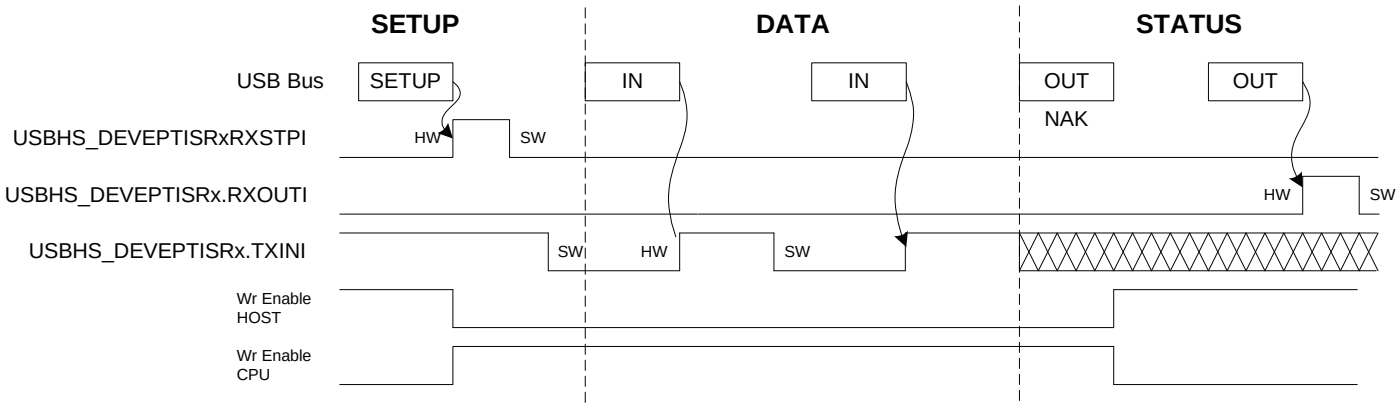
Figure 37-9. Control Write



Control Read

Figure 37-10 shows a control read transaction. The USBHS has to manage the simultaneous write requests from the CPU and the USB host.

Figure 37-10. Control Read



A NAK handshake is always generated on the first status stage command.

When the controller detects the status stage, all data written by the CPU is lost and clearing USBHS_DEVEPTISRx.TXINI has no effect.

The user checks if the transmission or the reception is complete.

The OUT retry is always ACKed. This reception sets USBHS_DEVEPTISRx.RXOUTI and USBHS_DEVEPTISRx.TXINI. Handle this with the following software algorithm:

```
set TXINI
```


Downloaded from Arrow.com.

```
wait for RXOUTI OR TXINI
if RXOUTI, then clear bit and return
if TXINI, then continue
```

Once the OUT status stage has been received, the USBHS waits for a SETUP request. The SETUP request has priority over any other request and has to be ACKed. This means that any other bit should be cleared and the FIFO reset when a SETUP is received.

The user has to consider that the byte counter is reset when a zero-length OUT packet is received.

37.5.2.12 Management of IN Endpoints

Overview

IN packets are sent by the USB device controller upon IN requests from the host. All data which acknowledges or not the bank can be written when it is full.

The endpoint must be configured first.

The USBHS_DEVEPTISRx.TXINI bit is set at the same time as USBHS_DEVEPTIMRx.FIFOCON when the current bank is free. This triggers a PEP_x interrupt if the Transmitted IN Data Interrupt Enable (USBHS_DEVEPTIMRx.TXINE) bit is one.

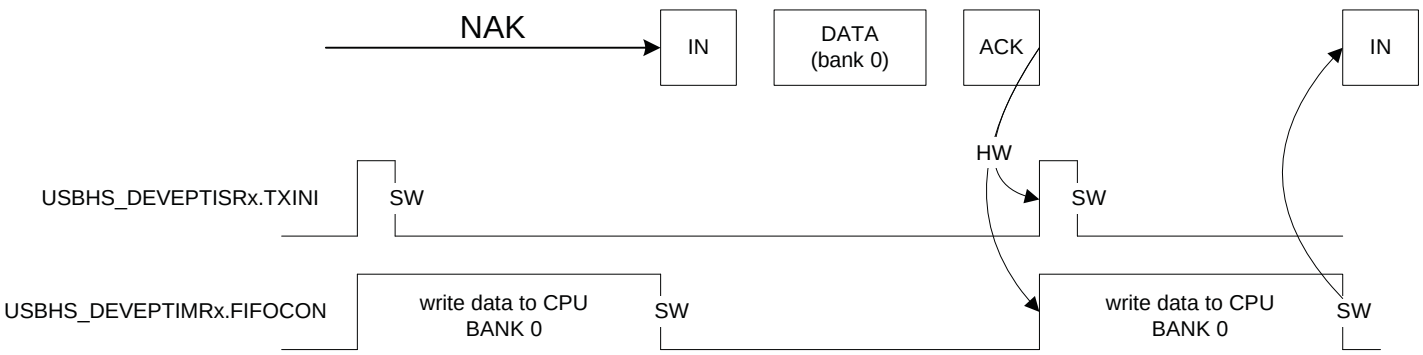
USBHS_DEVEPTISRx.TXINI is cleared by software (by writing a one to the Transmitted IN Data Interrupt Clear bit (USBHS_DEVEPTIDRx.TXINIC) to acknowledge the interrupt, which has no effect on the endpoint FIFO.

The user then writes into the FIFO and writes a one to the FIFO Control Clear (USBHS_DEVEPTIDRx.FIFOCONC) bit to clear the USBHS_DEVEPTIMRx.FIFOCON bit. This allows the USBHS to send the data. If the IN endpoint is composed of multiple banks, this also switches to the next bank. The USBHS_DEVEPTISRx.TXINI and USBHS_DEVEPTIMRx.FIFOCON bits are updated in accordance with the status of the next bank.

USBHS_DEVEPTISRx.TXINI is always cleared before clearing USBHS_DEVEPTIMRx.FIFOCON.

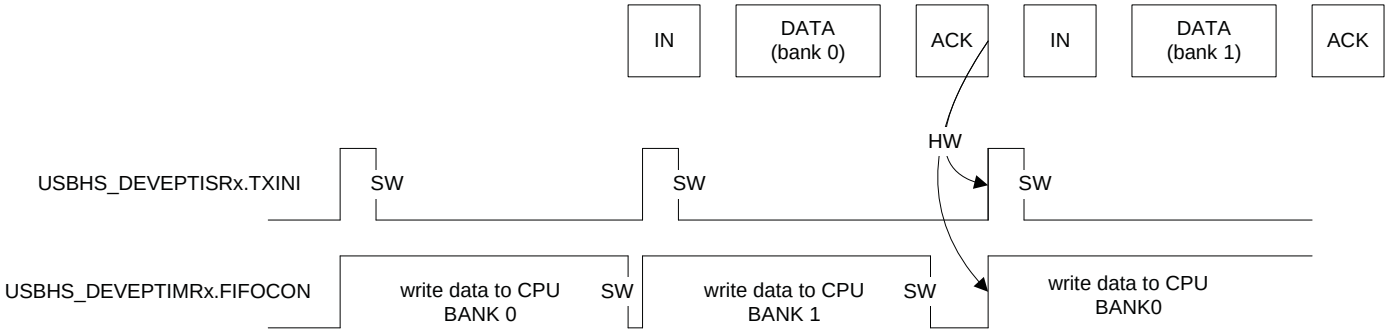
The USBHS_DEVEPTISRx.RWALL bit is set when the current bank is not full, i.e., when the software can write further data into the FIFO.

Figure 37-11. Example of an IN Endpoint with one Data Bank



Downloaded from Arrow.com.

Figure 37-12. Example of an IN Endpoint with two Data Banks



Downloaded from Arrow.com.

Detailed Description

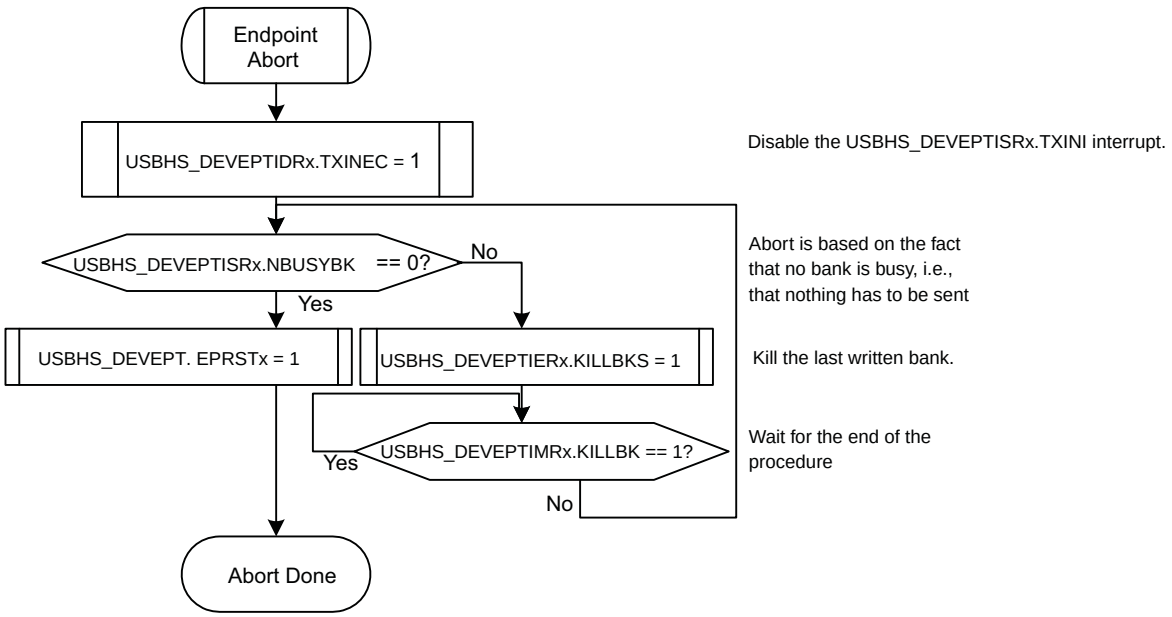
The data is written as follows:

- When the bank is empty, USBHS_DEVEPTISRx.TXINI and USBHS_DEVEPTIMRx.FIFOCON are set, which triggers a PEP_x interrupt if USBHS_DEVEPTIMRx.TXINE = 1.
- The user acknowledges the interrupt by clearing USBHS_DEVEPTISRx.TXINI.
- The user writes the data into the current bank by using the USB Pipe/Endpoint nFIFO Data (USBFIFOnDATA) register, until all the data frame is written or the bank is full (in which case USBHS_DEVEPTISRx.RWALL is cleared and the Byte Count (USBHS_DEVEPTISRx.BYCT) field reaches the endpoint size).
- The user allows the controller to send the bank and switches to the next bank (if any) by clearing USBHS_DEVEPTIMRx.FIFOCON.

If the endpoint uses several banks, the current one can be written while the previous one is being read by the host. Then, when the user clears USBHS_DEVEPTIMRx.FIFOCON, the following bank may already be free and USBHS_DEVEPTISRx.TXINI is set immediately.

An “Abort” stage can be produced when a zero-length OUT packet is received during an IN stage of a control or isochronous IN transaction. The Kill IN Bank (USBHS_DEVEPTIMRx.KILLBK) bit is used to kill the last written bank. The best way to manage this abort is to apply the algorithm represented in Figure 37-13.

Figure 37-13. Abort Algorithm



37.5.2.13 Management of OUT Endpoints

Overview

OUT packets are sent by the host. All data which acknowledges or not the bank can be read when it is empty.

The endpoint must be configured first.

The USBHS_DEVEPTISRx.RXOUTI bit is set at the same time as USBHS_DEVEPTIMRx.FIFOCON when the current bank is full. This triggers a PEP_x interrupt if the Received OUT Data Interrupt Enable (USBHS_DEVEPTIMRx.RXOUTE) bit is one.

USBHS_DEVEPTISRx.RXOUTI is cleared by software (by writing a one to the Received OUT Data Interrupt Clear (USBHS_DEVEPTICRx.RXOUTIC) bit to acknowledge the interrupt, which has no effect on the endpoint FIFO.

Downloaded from Arrow.com.

The user then reads from the FIFO and clears the USBHS_DEVEPTIMRx.FIFOCON bit to free the bank. If the OUT endpoint is composed of multiple banks, this also switches to the next bank. The USBHS_DEVEPTISRx.RXOUTI and USBHS_DEVEPTIMRx.FIFOCON bits are updated in accordance with the status of the next bank.

USBHS_DEVEPTISRx.RXOUTI is always cleared before clearing USBHS_DEVEPTIMRx.FIFOCON.

The USBHS_DEVEPTISRx.RWALL bit is set when the current bank is not empty, i.e., when the software can read further data from the FIFO.

Figure 37-14. Example of an OUT Endpoint with one Data Bank

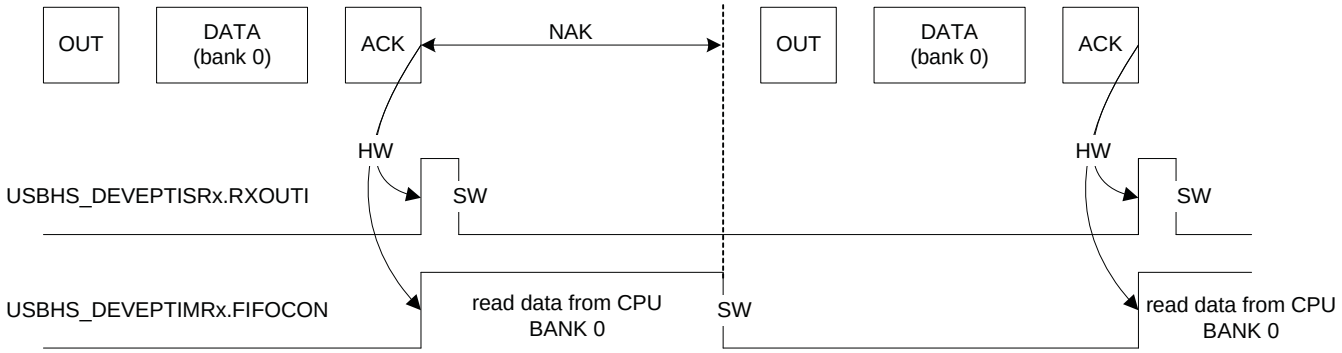
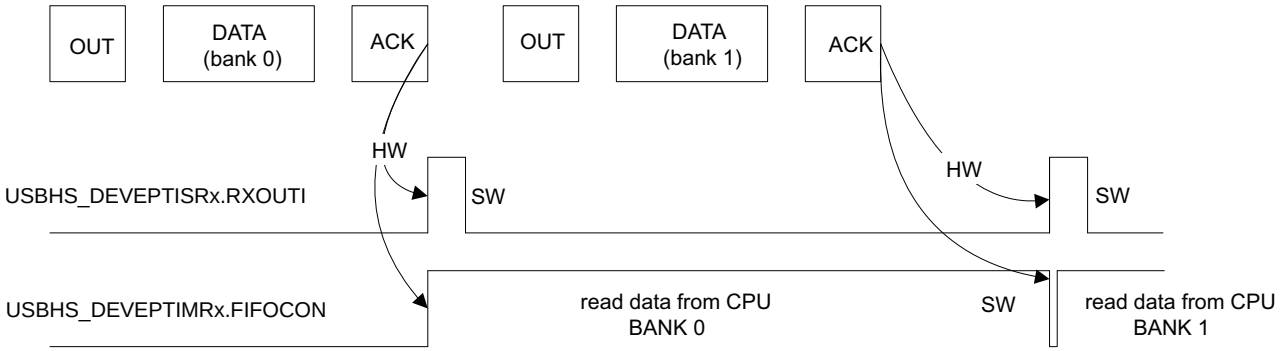


Figure 37-15. Example of an OUT Endpoint with two Data Banks



Downloaded from Arrow.com.

Detailed Description

The data is read as follows:

- When the bank is full, USBHS_DEVEPTISRx.RXOUTI and USBHS_DEVEPTIMRx.FIFOCON are set, which triggers a PEP_x interrupt if USBHS_DEVEPTIMRx.RXOUTE = 1.
- The user acknowledges the interrupt by writing a one to USBHS_DEVEPTICRx.RXOUTIC in order to clear USBHS_DEVEPTISRx.RXOUTI.
- The user can read the byte count of the current bank from USBHS_DEVEPTISRx.BYCT to know how many bytes to read, rather than polling USBHS_DEVEPTISRx.RWALL.
- The user reads the data from the current bank by using the USBFIFOnDATA register, until all the expected data frame is read or the bank is empty (in which case USBHS_DEVEPTISRx.RWALL is cleared and USBHS_DEVEPTISRx.BYCT reaches zero).
- The user frees the bank and switches to the next bank (if any) by clearing USBHS_DEVEPTIMRx.FIFOCON.

If the endpoint uses several banks, the current one can be read while the following one is being written by the host. Then, when the user clears USBHS_DEVEPTIMRx.FIFOCON, the following bank can already be read and USBHS_DEVEPTISRx.RXOUTI is set immediately.

In High-speed mode, the PING and NYET protocols are handled by the USBHS.

- For a single bank, a NYET handshake is always sent to the host (on Bulk-out transaction) to indicate that the current packet is acknowledged but there is no room for the next one.
- For a double bank, the USBHS responds to the OUT/DATA transaction with an ACK handshake when the endpoint accepted the data successfully and has room for another data payload (the second bank is free).

37.5.2.14 Underflow

This error only exists for isochronous IN/OUT endpoints. It sets the Underflow Interrupt (USBHS_DEVEPTISRx.UNDERFI) bit, which triggers a PEP_x interrupt if the Underflow Interrupt Enable (USBHS_DEVEPTIMRx.UNDERFE) bit is one.

- An underflow can occur during the IN stage if the host attempts to read from an empty bank. A zero-length packet is then automatically sent by the USBHS.
- An underflow cannot occur during the OUT stage on a CPU action, since the user may only read if the bank is not empty (USBHS_DEVEPTISRx.RXOUTI = 1 or USBHS_DEVEPTISRx.RWALL = 1).
- An underflow can also occur during the OUT stage if the host sends a packet while the bank is already full. Typically, the CPU is not fast enough. The packet is lost.
- An underflow cannot occur during the IN stage on a CPU action, since the user may only write if the bank is not full (USBHS_DEVEPTISRx.TXINI = 1or USBHS_DEVEPTISRx.RWALL = 1).

37.5.2.15 Overflow

This error exists for all endpoint types. It sets the Overflow interrupt (USBHS_DEVEPTISRx.OVERFI) bit, which triggers a PEP_x interrupt if the Overflow Interrupt Enable (USBHS_DEVEPTIMRx.OVERFE) bit is one.

- An overflow can occur during the OUT stage if the host attempts to write into a bank which is too small for the packet. The packet is acknowledged and the USBHS_DEVEPTISRx.RXOUTI bit is set as if no overflow had occurred. The bank is filled with all the first bytes of the packet that fit in.
- An overflow cannot occur during the IN stage on a CPU action, since the user may only write if the bank is not full (USBHS_DEVEPTISRx.TXINI = 1 or USBHS_DEVEPTISRx.RWALL = 1).

37.5.2.16 HB IsoIn Error

This error only exists for high-bandwidth isochronous IN endpoints.

At the end of the microframe, if at least one packet has been sent to the host and fewer banks than expected have been validated (by clearing the USBHS_DEVEPTIMRx.USBHS_DEVEPTIMRx.FIFOCON) for this microframe, it

Downloaded from Arrow.com.

sets the USBHS_DEVEPTISRx.HBISOINERRORI bit, which triggers a PEP_x interrupt if the High Bandwidth Isochronous IN Error Interrupt Enable (HBISOINERRORE) bit is one.

For example, if the Number of Transactions per MicroFrame for Isochronous Endpoint (NBTRANS) field in USBHS_DEVEPTCFGx is three (three transactions per microframe), only two banks are filled by the CPU (three expected) for the current microframe. Then, the HBISOINERRI interrupt is generated at the end of the microframe. Note that an UNDERFI interrupt is also generated (with an automatic zero-length-packet), except in the case of a missing IN token.

37.5.2.17 HB IsoFlush

This error only exists for high-bandwidth isochronous IN endpoints.

At the end of the microframe, if at least one packet has been sent to the host and there is a missing IN token during this microframe, the bank(s) destined to this microframe is/are flushed out to ensure a good data synchronization between the host and the device.

For example, if NBTRANS is three (three transactions per microframe) and if only the first IN token (among three) is well received by the USBHS, the last two banks are discarded.

37.5.2.18 CRC Error

This error only exists for isochronous OUT endpoints. It sets the CRC Error Interrupt (USBHS_DEVEPTISRx.CRCERRI) bit, which triggers a PEP_x interrupt if the CRC Error Interrupt Enable (USBHS_DEVEPTIMRx.CRCERRE) bit is one.

A CRC error can occur during the OUT stage if the USBHS detects a corrupted received packet. The OUT packet is stored in the bank as if no CRC error had occurred (USBHS_DEVEPTISRx.RXOUTI is set).

37.5.2.19 Interrupts

See the structure of the USB device interrupt system on Figure 37-3 on page 574.

There are two kinds of device interrupts: processing, i.e., their generation is part of the normal processing, and exception, i.e., errors (not related to CPU exceptions).

Global Interrupts

The processing device global interrupts are:

- Suspend (USBHS_DEVISR.SUSP)
- Start of Frame (USBHS_DEVISR.SOF) interrupt with no frame number CRC error - the Frame Number CRC Error (USBHS_DEVFNUM.FNCERR) bit is zero.
- Micro Start of Frame (USBHS_DEVISR.MSOF) with no CRC error
- End of Reset (USBHS_DEVISR.EORST)
- Wake-Up (USBHS_DEVISR.WAKEUP)
- End of Resume (USBHS_DEVISR.EORSM)
- Upstream Resume (USBHS_DEVISR.UPRSM)
- Endpoint x (USBHS_DEVISR.PEP_x)
- DMA Channel x (USBHS_DEVISR.DMA_x)

The exception device global interrupts are:

- Start of Frame (USBHS_DEVISR.SOF) with a frame number CRC error (USBHS_DEVFNUM.FNCERR = 1)
- Micro Start of Frame (USBHS_DEVFNUM.FNCERR.MSOF) with a CRC error

Endpoint Interrupts

The processing device endpoint interrupts are:

- Transmitted IN Data (USBHS_DEVEPTISRx.TXINI)
- Received OUT Data (USBHS_DEVEPTISRx.RXOUTI)

Downloaded from Arrow.com.

- Received SETUP (USBHS_DEVEPTISRx.RXSTPI)
- Short Packet (USBHS_DEVEPTISRx.SHORTPACKET)
- Number of Busy Banks (USBHS_DEVEPTISRx.NBUSYBK)
- Received OUT Isochronous Multiple Data (DTSEQ=MDATA & USBHS_DEVEPTISRx.RXOUTI)
- Received OUT Isochronous DataX (DTSEQ=DATAX & USBHS_DEVEPTISRx.RXOUTI)

The exception device endpoint interrupts are:

- Underflow (USBHS_DEVEPTISRx.UNDERFI)
- NAKed OUT (USBHS_DEVEPTISRx.NAKOUTI)
- High-Bandwidth Isochronous IN Error (USBHS_DEVEPTISRx.HBISOINERRI)
- NAKed IN (USBHS_DEVEPTISRx.NAKINI)
- High-Bandwidth Isochronous IN Flush error (USBHS_DEVEPTISRx.HBISOFLUSHI)
- Overflow (USBHS_DEVEPTISRx.OVERFI)
- STALLed (USBHS_DEVEPTISRx.STALLEDI)
- CRC Error (USBHS_DEVEPTISRx.CRCERRI)
- Transaction Error (USBHS_DEVEPTISRx.ERRORTRANS)

DMA Interrupts

The processing device DMA interrupts are:

- End of USB Transfer Status (USBHS_DEVDMASTATUSx.END_TR_ST)
- End of Channel Buffer Status (USBHS_DEVDMASTATUSx.END_BF_ST)
- Descriptor Loaded Status (USBHS_DEVDMASTATUSx.DESC_LDST)

There is no exception device DMA interrupt.

37.5.2.20 Test Modes

When written to one, the USBHS_DEVCTRL.TSTPCKT bit switches the USB device controller to a “Test-packet” mode:

The transceiver repeatedly transmits the packet stored in the current bank. USBHS_DEVCTRL.TSTPCKT must be written to zero to exit the Test-packet mode. The endpoint is reset by software after a Test-packet mode.

This enables the testing of rise and falling times, eye patterns, jitter, and any other dynamic waveform specifications.

The flow control used to send the packets is as follows:

- USBHS_DEVCTRL.TSTPCKT=1;
- Store data in an endpoint bank
- Write a zero to the USBHS_DEVEPTIDRx.FIFOCON bit

To stop the Test-packet mode, write a zero to the USBHS_DEVCTRL.TSTPCKT bit.

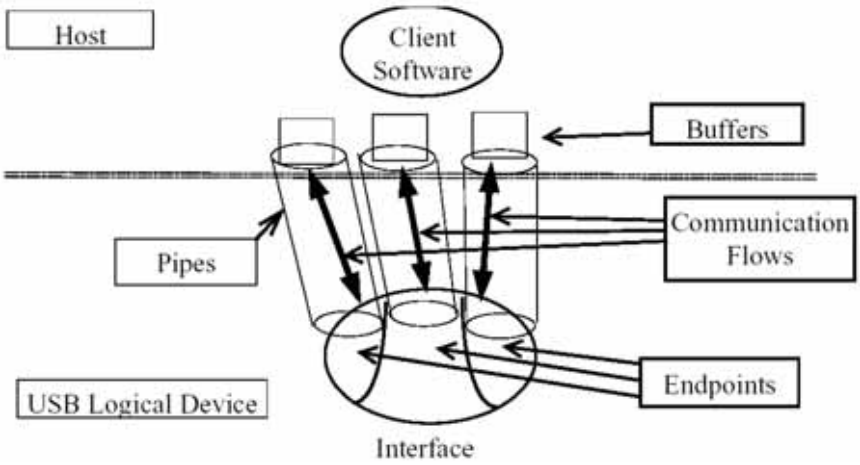
Downloaded from Arrow.com.

37.5.3 USB Host Operation

37.5.3.1 Description of Pipes

For the USBHS in Host mode, the term “pipe” is used instead of “endpoint” (used in Device mode). A host pipe corresponds to a device endpoint, as described in Figure 37-16 (from the USB Specification).

Figure 37-16. USB Communication Flow

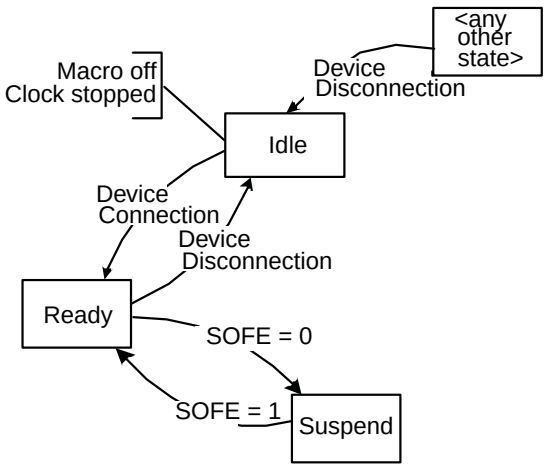


In Host mode, the USBHS associates a pipe to a device endpoint, considering the device configuration descriptors.

37.5.3.2 Power-On and Reset

Figure 37-17 describes the USBHS Host mode main states.

Figure 37-17. Host Mode Main States



After a hardware reset, the USBHS Host mode is in the Reset state.

When the USBHS is enabled (USBHS_CTRL.USBE = 1) in Host mode (USBHS_CTRL.UIMOD = 0), it goes to the Idle state. In this state, the controller waits for a device connection with a minimal power consumption. The USB pad should be in the Idle state. Once a device is connected, the USBHS enters the Ready state, which does not require the USB clock to be activated.

Downloaded from Arrow.com.

The controller enters the Suspend state when the USB bus is in a “Suspend” state, i.e., when the Host mode does not generate the “Start of Frame (SOF)”. In this state, the USB consumption is minimal. The Host mode exits the Suspend state when starting to generate the SOF over the USB line.

37.5.3.3 Device Detection

A device is detected by the USBHS Host mode when D+ or D- is no longer tied low, i.e., when the device D+ or D- pull-up resistor is connected. The bit USBHS_SFR.VBUSRQS must be set to ‘1’ to enable this detection.

Note: The VBUS supply is not managed by the USBHS interface. It must be generated on-board.

The device disconnection is detected by the host controller when both D+ and D- are pulled down.

37.5.3.4 USB Reset

The USBHS sends a USB bus reset when the user writes a one to the Send USB Reset bit in the Host General Control register (USBHS_HSTCTRL.RESET). The USB Reset Sent Interrupt bit in the Host Global Interrupt Status register (USBHS_HSTISR.RSTI) is set when the USB reset has been sent. In this case, all pipes are disabled and de-allocated.

If the bus was previously in a “Suspend” state (the Start of Frame Generation Enable (USBHS_HSTCTRL.SOFE) bit is zero), the USBHS automatically switches to the “Resume” state, the Host Wake-Up Interrupt (USBHS_HSTISR.HWUPI) bit is set and the USBHS_HSTCTRL.SOFE bit is set in order to generate SOFs or micro SOFs immediately after the USB reset.

At the end of the reset, the user should check the USBHS_SR.SPEED field to know the speed running according to the peripheral capability (LS.FS/HS).

37.5.3.5 Pipe Reset

A pipe can be reset at any time by writing a one to the Pipe x Reset (USBHS_HSTPIP.PRSTx) bit. This is recommended before using a pipe upon hardware reset or when a USB bus reset has been sent. This resets:

- the internal state machine of the pipe,
- the receive and transmit bank FIFO counters,
- all the registers of the pipe (USBHS_HSTPIPCFGx, USBHS_HSTPIPISRx, USBHS_HSTPIPIMRx), except its configuration (USBHS_HSTPIPCFGx.ALLOC, USBHS_HSTPIPCFGx.PBK, USBHS_HSTPIPCFGx.PSIZE, USBHS_HSTPIPCFGx.PTOKEN, USBHS_HSTPIPCFGx.PTYPE, USBHS_HSTPIPCFGx.PEPNUM, USBHS_HSTPIPCFGx.INTFRQ) and its Data Toggle Sequence field (USBHS_HSTPIPISRx.DTSEQ).

The pipe configuration remains active and the pipe is still enabled.

The pipe reset may be associated with a clear of the data toggle sequence. This can be achieved by setting the Reset Data Toggle bit in the Pipe x Control register (USBHS_HSTPIPIMRx.RSTDT) (by writing a one to the Reset Data Toggle Set bit in the Pipe x Control Set register (USBHS_HSTPIPIERx.RSTDTS)).

In the end, the user has to write a zero to the USBHS_HSTPIP.PRSTx bit to complete the reset operation and to start using the FIFO.

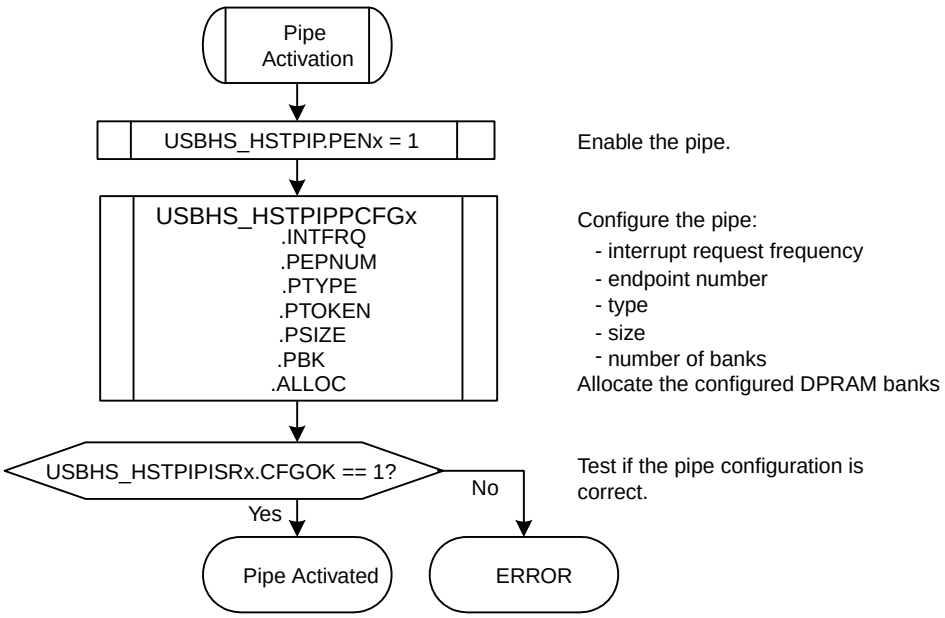
37.5.3.6 Pipe Activation

The pipe is maintained inactive and reset (see Section 37.5.3.5 for more details) as long as it is disabled (USBHS_HSTPIP.PENx = 0). The Data Toggle Sequence field (USBHS_HSTPIPISRx.DTSEQ) is also reset.

The algorithm represented on Figure 37-18 must be followed in order to activate a pipe.

Downloaded from Arrow.com.

Figure 37-18. Pipe Activation Algorithm



As long as the pipe is not correctly configured (USBHS_HSTPIPISRx.CFGOK = 0), the controller cannot send packets to the device through this pipe.

The USBHS_HSTPIPISRx.CFGOK bit is only set if the configured size and number of banks are correct as compared to their maximal allowed values for the pipe (see Table 37-1 on page 570) and to the maximal FIFO size (i.e., the DPRAM size).

See Section 37.5.1.5 for more details about DPRAM management.

Once the pipe is correctly configured (USBHS_HSTPIPISRx.CFGOK = 1), only the USBHS_HSTPIPCFGx.PTOKEN and USBHS_HSTPIPCFGx.INTFRQ fields can be written by software. USBHS_HSTPIPCFGx.INTFRQ is meaningless for non-interrupt pipes.

When starting an enumeration, the user gets the device descriptor by sending a GET_DESCRIPTOR USB request. This descriptor contains the maximal packet size of the device default control endpoint (bMaxPacketSize0) and the user reconfigures the size of the default control pipe with this size parameter.

37.5.3.7 Address Setup

Once the device has answered the first host requests with the default device address 0, the host assigns a new address to the device. The host controller has to send a USB reset to the device and to send a SET_ADDRESS (addr) SETUP request with the new address to be used by the device. Once this SETUP transaction is over, the user writes the new address into the USB Host Address for Pipe x field in the USB Host Device Address register (HSTADDR.HSTADDRPx). All the following requests on all pipes are then performed using this new address.

When the host controller sends a USB reset, the HSTADDRPx field is reset by hardware and the following host requests are performed using the default device address 0.

37.5.3.8 Remote Wake-up

The controller Host mode enters the Suspend state when the USBHS_HSTCTRL.SOFE bit is written to zero. No more “Start of Frame” is sent on the USB bus and the USB device enters the Suspend state 3 ms later.

The device awakes the host by sending an Upstream Resume (Remote Wake-Up feature). When the host controller detects a non-idle state on the USB bus, it sets the Host Wake-Up interrupt (USBHS_HSTISR.HWUPI) bit. If the non-idle bus state corresponds to an Upstream Resume (K state), the Upstream Resume Received Interrupt (USBHS_HSTISR.RXRSMI) bit is set. The user has to generate a Downstream Resume within 1 ms and

Downloaded from Arrow.com.

for at least 20 ms by writing a one to the Send USB Resume (USBHS_HSTCTRL.RESUME) bit. It is mandatory to write a one to USBHS_HSTCTRL.SOFE before writing a one to USBHS_HSTCTRL.RESUME to enter the Ready state, otherwise USBHS_HSTCTRL.RESUME has no effect.

37.5.3.9 Management of Control Pipes

A control transaction is composed of three stages:

- SETUP
- Data (IN or OUT)
- Status (OUT or IN)

The user has to change the pipe token according to each stage.

For the control pipe only, each token is assigned a specific initial data toggle sequence:

- SETUP: Data0
- IN: Data1
- OUT: Data1

37.5.3.10 Management of IN Pipes

IN packets are sent by the USB device controller upon IN requests from the host. All data which acknowledges or not the bank can be read when it is empty.

The pipe must be configured first.

When the host requires data from the device, the user has to first select the IN Request mode with the IN Request Mode bit in the Pipe x IN Request register (USBHS_HSTPIPINRQx.INMODE):

- When USBHS_HSTPIPINRQx.INMODE = 0, the USBHS performs (INRQ + 1) IN requests before freezing the pipe.
- When USBHS_HSTPIPINRQx.INMODE = 1, the USBHS performs IN requests endlessly when the pipe is not frozen by the user.

The generation of IN requests starts when the pipe is unfrozen (the Pipe Freeze (USBHS_HSTPIPIMRx.PFREEZE) field in USBHS_HSTPIPIMRx is zero).

The Received IN Data Interrupt (USBHS_HSTPIPISRx.RXINI) bit is set at the same time as the FIFO Control (USBHS_HSTPIPIMRx.FIFOCON) bit when the current bank is full. This triggers a PEP_x interrupt if the Received IN Data Interrupt Enable (USBHS_HSTPIPIMRx.RXINE) bit is one.

USBHS_HSTPIPISRx.RXINI is cleared by software (by writing a one to the Received IN Data Interrupt Clear bit in the Host Pipe x Clear register (USBHS_HSTPIPIDRx.RXINIC)) to acknowledge the interrupt, which has no effect on the pipe FIFO.

The user then reads from the FIFO and clears the USBHS_HSTPIPIMRx.FIFOCON bit (by writing a one to the FIFO Control Clear (USBHS_HSTPIPIDRx.FIFOCONC) bit) to free the bank. If the IN pipe is composed of multiple banks, this also switches to the next bank. The USBHS_HSTPIPISRx.RXINI and USBHS_HSTPIPIMRx.FIFOCON bits are updated in accordance with the status of the next bank.

USBHS_HSTPIPISRx.RXINI is always cleared before clearing USBHS_HSTPIPIMRx.FIFOCON.

The Read/Write Allowed (USBHS_HSTPIPISRx.RWALL) bit is set when the current bank is not empty, i.e., when the software can read further data from the FIFO.

Downloaded from Arrow.com.

Figure 37-19. Example of an IN Pipe with one Data Bank

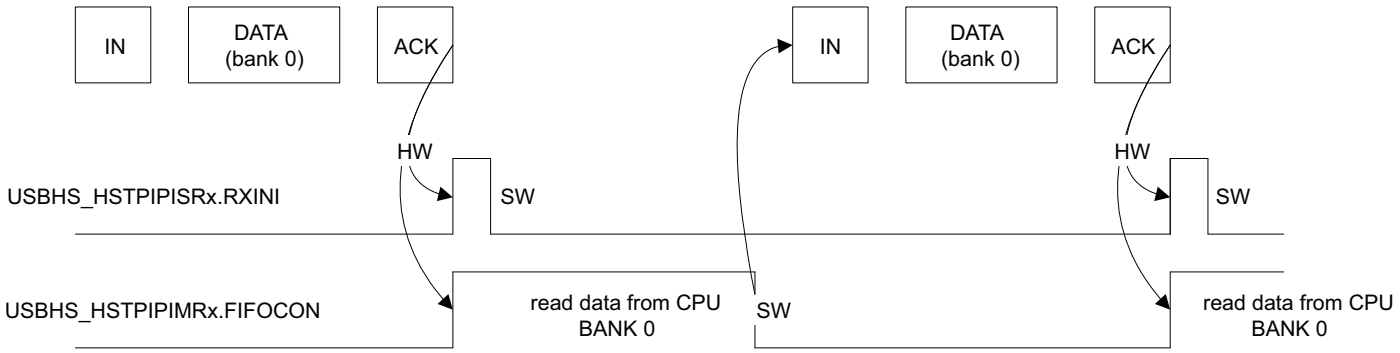
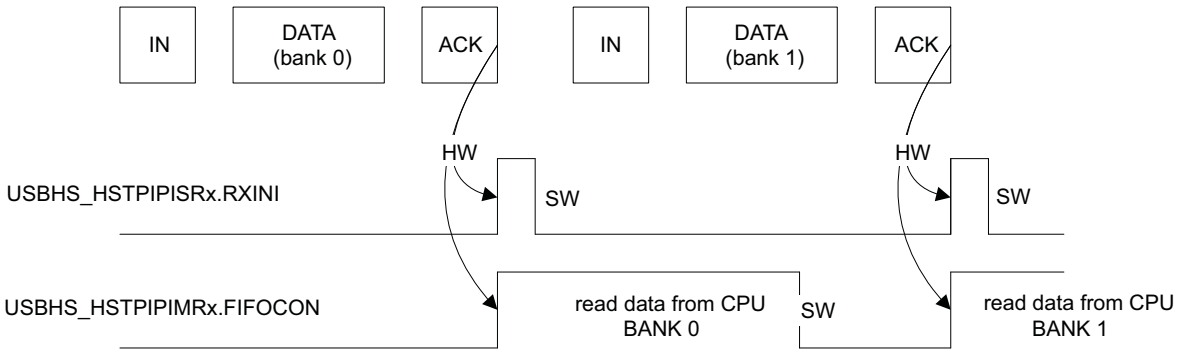


Figure 37-20. Example of an IN Pipe with two Data Banks



37.5.3.11 Management of OUT Pipes

OUT packets are sent by the host. All data which acknowledges or not the bank can be written when it is full.

The pipe must be configured and unfrozen first.

The Transmitted OUT Data Interrupt (USBHS_HSTPIPISRx.TXOUTI) bit is set at the same time as USBHS_HSTPIPIMRx.FIFOCON when the current bank is free. This triggers a PEP_x interrupt if the Transmitted OUT Data Interrupt Enable (USBHS_HSTPIPIMRx.TXOUTE) bit is one.

USBHS_HSTPIPISRx.TXOUTI is cleared by software (by writing a one to the Transmitted OUT Data Interrupt Clear (USBHS_HSTPIPIDRx.TXOUTIC) bit to acknowledge the interrupt, which has no effect on the pipe FIFO.

The user then writes into the FIFO and clears the USBHS_HSTPIPIDRx.FIFOCON bit to allow the USBHS to send the data. If the OUT pipe is composed of multiple banks, this also switches to the next bank. The USBHS_HSTPIPISRx.TXOUTI and USBHS_HSTPIPIMRx.FIFOCON bits are updated in accordance with the status of the next bank.

USBHS_HSTPIPISRx.TXOUTI is always cleared before clearing USBHS_HSTPIPIMRx.FIFOCON.

The USBHS_HSTPIPISRx.RWALL bit is set when the current bank is not full, i.e., when the software can write further data into the FIFO.

Notes:
1. If the user decides to switch to the Suspend state (by writing a zero to the USBHS_HSTCTRL.SOFE bit) while a bank is ready to be sent, the USBHS automatically exits this state and the bank is sent.
2. In High-speed operating mode, the host controller automatically manages the PING protocol to maximize the USB bandwidth. The user can tune the PING protocol by handling the Ping Enable (PINGEN) bit and the bInterval

Downloaded from Arrow.com.

Parameter for the Bulk-Out/Ping Transaction (BINTERVAL) field in USBHS_HSTPIPCFGx. See Section 37.6.43 for more details.

Figure 37-21. Example of an OUT Pipe with one Data Bank

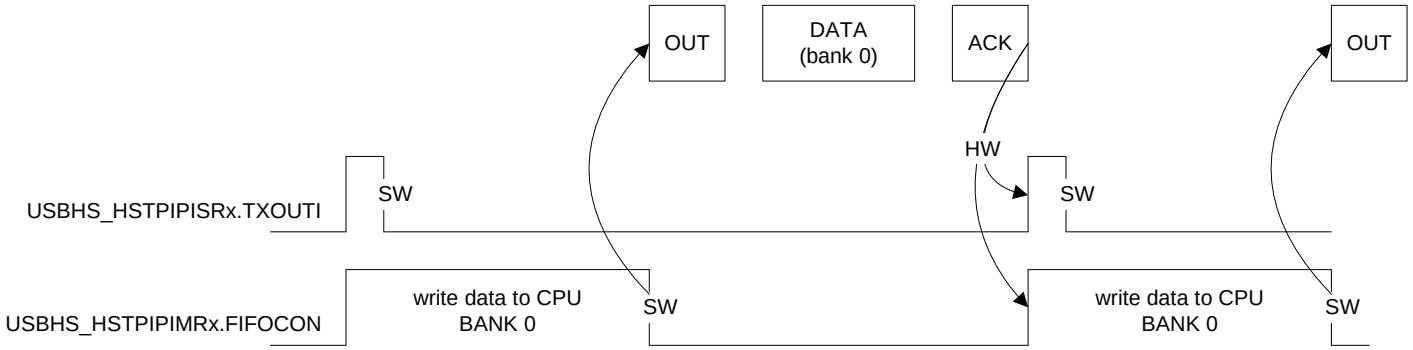


Figure 37-22. Example of an OUT Pipe with two Data Banks and no Bank Switching Delay

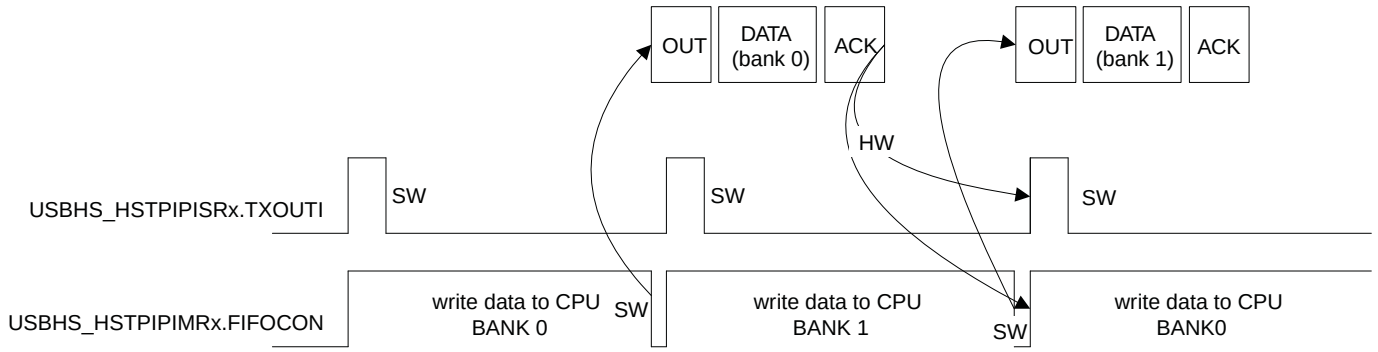
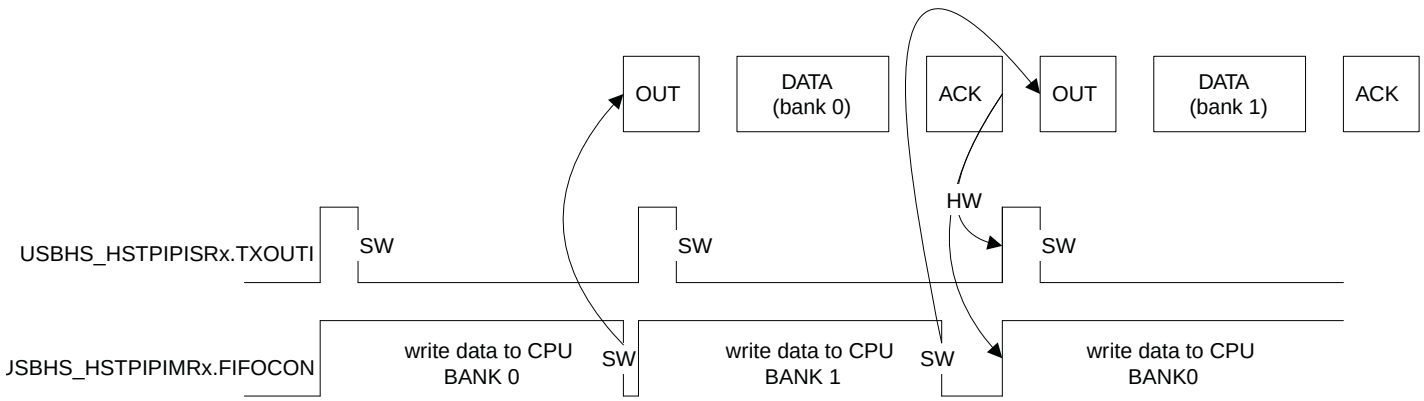


Figure 37-23. Example of an OUT Pipe with two Data Banks and a Bank Switching Delay



Downloaded from Arrow.com.

37.5.3.12 CRC Error

This error exists only for isochronous IN pipes. It sets the CRC Error Interrupt (USBHS_HSTPIPISRx.CRCERRI) bit, which triggers a PEP_x interrupt if then the CRC Error Interrupt Enable (USBHS_HSTPIPIMRx.CRCERRE) bit is one.

A CRC error can occur during IN stage if the USBHS detects a corrupted received packet. The IN packet is stored in the bank as if no CRC error had occurred (USBHS_HSTPIPISRx.RXINI is set).

37.5.3.13 Interrupts

See the structure of the USB host interrupt system on Figure 37-3 on page 574.

There are two kinds of host interrupts: processing, i.e., their generation is part of the normal processing, and exception, i.e., errors (not related to CPU exceptions).

Global Interrupts

The processing host global interrupts are:

- Device Connection (USBHS_HSTISR.DCONNI)
- Device Disconnection (USBHS_HSTISR.DDISCI)
- USB Reset Sent (USBHS_HSTISR.RSTI)
- Downstream Resume Sent (USBHS_HSTISR.RSMEDI)
- Upstream Resume Received (USBHS_HSTISR.RXRSMI)
- Host Start of Frame (USBHS_HSTISR.HSOFI)
- Host Wake-Up (USBHS_HSTISR.HWUPI)
- Pipe x (USBHS_HSTISR.PEP_x)
- DMA Channel x (USBHS_HSTISR.DMAxINT)

There is no exception host global interrupt.

Pipe Interrupts

The processing host pipe interrupts are:

- Received IN Data (USBHS_HSTPIPISRx.RXINI)
- Transmitted OUT Data (USBHS_HSTPIPISRx.TXOUTI)
- Transmitted SETUP (USBHS_HSTPIPISRx.TXSTPI)
- Short Packet (USBHS_HSTPIPISRx.SHORTPACKETI)
- Number of Busy Banks (USBHS_HSTPIPISRx.NBUSYBK)

The exception host pipe interrupts are:

- Underflow (USBHS_HSTPIPISRx.UNDERFI)
- Pipe Error (USBHS_HSTPIPISRx.PERRI)
- NAKed (USBHS_HSTPIPISRx.NAKEDI)
- Overflow (USBHS_HSTPIPISRx.OVERFI)
- Received STALLed (USBHS_HSTPIPISRx.RXSTALLDI)
- CRC Error (USBHS_HSTPIPISRx.CRCERRI)

DMA Interrupts

The processing host DMA interrupts are:

- The End of USB Transfer Status (USBHS_HSTDMASTATUSx.END_TR_ST)
- The End of Channel Buffer Status (USBHS_HSTDMASTATUSx.END_BF_ST)
- The Descriptor Loaded Status (USBHS_HSTDMASTATUSx.DESC_LDST)

There is no exception host DMA interrupt.

Downloaded from Arrow.com.

37.5.4 USB DMA Operation

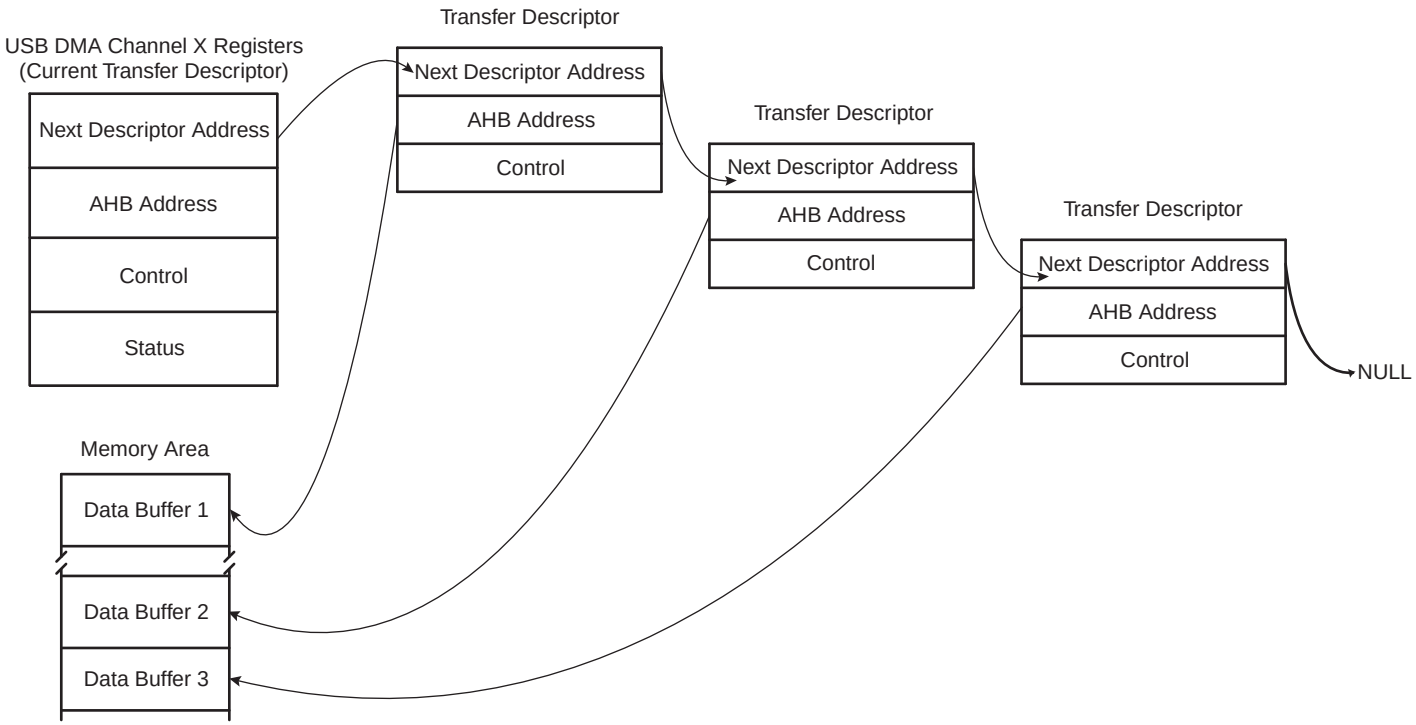
USB packets of any length may be transferred when required by the USBHS. These transfers always feature sequential addressing. Such characteristics mean that in case of high USBHS throughput, both AHB ports benefit from “incrementing burst of unspecified length” since the average access latency of AHB slaves can then be reduced.

The DMA uses word “incrementing burst of unspecified length” of up to 256 beats for both data transfers and channel descriptor loading. A burst may last on the AHB busses for the duration of a whole USB packet transfer, unless otherwise broken by the AHB arbitration or the AHB 1-Kbyte boundary crossing.

Packet data AHB bursts may be locked on a DMA buffer basis for drastic overall AHB bus bandwidth performance boost with paged memories. This prevents large AHB bursts from being broken in case of conflict with other AHB bus masters, thus avoiding access latencies due to memory row changes. This means up to 128 words single cycle unbroken AHB bursts for bulk pipes/endpoints and 256 words single cycle unbroken bursts for isochronous pipes/endpoints. This maximal burst length is then controlled by the lowest programmed USB Pipe/Endpoint Size (USBHS_HSTPIPCFGx.PSIZE / USBHS_DEVEPTCFGx.EPSIZE) and the Buffer Byte Length (USBHS_HSTDMACONTROLx.BUFF_LENGTH / USBHS_DEVDMACONTROLx.BUFF_LENGTH) fields.

The USBHS average throughput can reach nearly 480 Mbps. Its average access latency decreases as burst length increases due to the zero wait-state side effect of unchanged pipe/endpoint. Word access allows reducing the AHB bandwidth required for the USB by four, as compared to native byte access. If at least 0 wait-state word burst capability is also provided by the other DMA AHB bus slaves, each DMA AHB bus needs less than 60% bandwidth allocation for full USB bandwidth usage at 33 MHz, and less than 30% at 66 MHz.

Figure 37-24. Example of a DMA Chained List



Downloaded from Arrow.com.

37.5.5 USB DMA Channel Transfer Descriptor

The DMA channel transfer descriptor is loaded from the memory. The following structures apply:

Offset 0:

- The address must be aligned: 0xXXXX0
- Next Descriptor Address Register: USBHS_xxxDMANXTDSCx

Offset 4:

- The address must be aligned: 0xXXXX4
- DMA Channelx Address Register: USBHS_xxxDMAADDRESSx

Offset 8:

- The address must be aligned: 0xXXXX8
- DMA Channelx Control Register: USBHS_xxxDMACONTROLx

To use the DMA channel transfer descriptor, fill the structures with the correct values (as described in the following pages), then write directly in USBHS_xxxDMANXTDSCx the address of the descriptor to be used first.

Then write 1 in the USBHS_xxxDMACONTROLx.LDNXT_DSC bit (load next channel transfer descriptor). The descriptor is automatically loaded upon pipe x / endpoint x request for packet transfer.

Downloaded from Arrow.com.

37.6 USB High-Speed (USBHS) User Interface

Table 37-4. Register Mapping

Offset	Register	Name	Access	Reset
0x0000	Device General Control Register	USBHS_DEVCTRL	Read/Write	0x00000100
0x0004	Device Global Interrupt Status Register	USBHS_DEVISR	Read-only	0x00000000
0x0008	Device Global Interrupt Clear Register	USBHS_DEVICR	Write-only	
0x000C	Device Global Interrupt Set Register	USBHS_DEVIFR	Write-only	
0x0010	Device Global Interrupt Mask Register	USBHS_DEVIMR	Read-only	0x00000000
0x0014	Device Global Interrupt Disable Register	USBHS_DEVIDR	Write-only	
0x0018	Device Global Interrupt Enable Register	USBHS_DEVIER	Write-only	
0x001C	Device Endpoint Register	USBHS_DEVEPT	Read/Write	0x00000000
0x0020	Device Frame Number Register	USBHS_DEVFNUM	Read-only	0x00000000
0x0100 + (n * 0x04) + 0x00	Device Endpoint Configuration Register	USBHS_DEVEPTCFG	Read/Write	0x00002000
0x0100 + (n * 0x04) + 0x30	Device Endpoint Status Register	USBHS_DEVEPTISR	Read-only	0x00000100
0x0100 + (n * 0x04) + 0x60	Device Endpoint Clear Register	USBHS_DEVEPTICR	Write-only	
0x0100 + (n * 0x04) + 0x90	Device Endpoint Set Register	USBHS_DEVEPTIFR	Write-only	
0x0100 + (n * 0x04) + 0x0C0	Device Endpoint Mask Register	USBHS_DEVEPTIMR	Read-only	0x00000000
0x0100 + (n * 0x04) + 0x0F0	Device Endpoint Enable Register	USBHS_DEVEPTIER	Write-only	
0x0100 + (n * 0x04) + 0x0120	Device Endpoint Disable Register	USBHS_DEVEPTIDR	Write-only	
0x0300 + (n * 0x10)+0x00	Device DMA Channel Next Descriptor Address Register	USBHS_DEVDMANXTDSC	Read/Write	0x00000000
0x0300 + (n * 0x10)+0x04	Device DMA Channel Address Register	USBHS_DEVDMAADDRESS	Read/Write	0x00000000
0x0300 + (n * 0x10)+0x08	Device DMA Channel Control Register	USBHS_DEVDMACONTROL	Read/Write	0x00000000
0x0300 + (n * 0x10)+0x0C	Device DMA Channel Status Register	USBHS_DEVDMASTATUS	Read/Write	0x00000000
0x0400	Host General Control Register	USBHS_HSTCTRL	Read/Write	0x00000000
0x0404	Host Global Interrupt Status Register	USBHS_HSTISR	Read-only	0x00000000
0x0408	Host Global Interrupt Clear Register	USBHS_HSTICR	Write-only	
0x040C	Host Global Interrupt Set Register	USBHS_HSTIFR	Write-only	
0x0410	Host Global Interrupt Mask Register	USBHS_HSTIMR	Read-only	0x00000000
0x0414	Host Global Interrupt Disable Register	USBHS_HSTIDR	Write-only	
0x0418	Host Global Interrupt Enable Register	USBHS_HSTIER	Write-only	
0x0041C	Host Pipe Register	USBHS_HSTPIP	Read/Write	0x00000000
0x0420	Host Frame Number Register	USBHS_HSTFNUM	Read/Write	0x00000000
0x0424	Host Address 1 Register	USBHS_HSTADDR1	Read/Write	0x00000000
0x0428	Host Address 2 Register	USBHS_HSTADDR2	Read/Write	0x00000000
0x042C	Host Address 3 Register	USBHS_HSTADDR3	Read/Write	0x00000000
0x0500 + (n * 0x04) + 0x00	Host Pipe Configuration Register	USBHS_HSTPIPCFG	Read/Write	0x00000000
0x0500 + (n * 0x04) + 0x30	Host Pipe Status Register	USBHS_HSTPIPISR	Read-only	0x00000000
0x0500 + (n * 0x04) + 0x60	Host Pipe Clear Register	USBHS_HSTPIPICR	Write-only	
0x0500 + (n * 0x04) + 0x90	Host Pipe Set Register	USBHS_HSTPIPIFR	Write-only	

Downloaded from Arrow.com.

Table 37-4. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x0500 + (n * 0x04) + 0xC0	Host Pipe Mask Register	USBHS_HSTPIPIMR	Read-only	0x00000000
0x0500 + (n * 0x04) + 0xF0	Host Pipe Enable Register	USBHS_HSTPIPIER	Write-only	
0x0500+ (n * 0x04) + 0x120	Host Pipe Disable Register	USBHS_HSTPIPIDR	Write-only	
0x0500+ (n * 0x04) + 0x150	Host Pipe IN Request Register	USBHS_HSTPIPINRQ	Read/Write	0x00000000
0x0500 + (n * 0x04) + 0x180	Host Pipe Error Register	USBHS_HSTPIPERR	Read/Write	0x00000000
0x0700 + (n * 0x10) + 0x00	Host DMA Channel Next Descriptor Address Register	USBHS_HSTDMANXTDSC	Read/Write	0x00000000
0x0700 + (n * 0x10) + 0x04	Host DMA Channel Address Register	USBHS_HSTDMAADDRESS	Read/Write	0x00000000
0x0700 + (n * 0x10) + 0x08	Host DMA Channel Control Register	USBHS_HSTDMACONTROL	Read/Write	0x00000000
0x0700 + (n * 0x10) + 0x0C	Host DMA Channel Status Register	USBHS_HSTDMASTATUS	Read/Write	0x00000000
0x0800	General Control Register	USBHS_CTRL	Read/Write	0x03004000
0x0804	General Status Register	USBHS_SR	Read-only	0x00000400
0x0808	General Status Clear Register	USBHS_SCR	Write-only	
0x080C	General Status Set Register	USBHS_SFR	Write-only	
0x0810 - 0x082C	Reserved	–	–	–

Downloaded from Arrow.com.

37.6.1 General Control Register

Name: USBHS_CTRL

Address: 0x40038800

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	UIMOD	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
USBE	FRZCLK	–	–	–	–	–	VBUSHWC
7	6	5	4	3	2	1	0
–	–	–	RDERRE	–	–	–	–

- **RDERRE: Remote Device Connection Error Interrupt Enable**

0: The Remote Device Connection Error Interrupt (USBHS_SR.RDERRI) is disabled.

1: The Remote Device Connection Error Interrupt (USBHS_SR.RDERRI) is enabled.

- **VBUSHWC: VBUS Hardware Control**

Must be set to ‘1’.

- **FRZCLK: Freeze USB Clock**

0: The clock inputs are enabled.

1: The clock inputs are disabled (the resume detection is still active). This reduces the power consumption. Unless explicitly stated, all registers then become read-only.

This bit can be written even if USBE = 0. Disabling the USBHS (by writing a zero to the USBE bit) does not reset this bit, but it freezes the clock inputs whatever its value.

- **USBE: USBHS Enable**

Writing a zero to this bit resets the USBHS, disables the USB transceiver, and disables the USBHS clock inputs. Unless explicitly stated, all registers then become read-only and are reset.

0: The USBHS is disabled.

1: The USBHS is enabled.

This bit can be written even if FRZCLK = 1

- **UIMOD: USBHS Mode**

0 (HOST): The module is in USB Host mode.

1 (DEVICE): The module is in USB Device mode.

This bit can be written even if USBE = 0 or FRZCLK = 1. Disabling the USBHS (by writing a zero to the USBE bit) does not reset this bit.

Downloaded from Arrow.com.

37.6.2 General Status Register

Name: USBHS_SR

Address: 0x40038804

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	CLKUSABLE	SPEED		–	–	–	–
7	6	5	4	3	2	1	0
-	–	–	RDERRI	–	–	–	–

- **RDERRI: Remote Device Connection Error Interrupt (Host mode only)**

0: Cleared when USBHS_SCR.RDERRIC = 1.

1: Set when an error occurs during the remote device connection. This triggers a USB interrupt if USBHS_CTRL.RDERRE = 1.

- **SPEED: Speed Status (Device mode only)**

This field is set according to the controller speed mode.

Value	Name	Description
0	FULL_SPEED	Full-Speed mode
1	HIGH_SPEED	High-Speed mode
2	LOW_SPEED	Low-Speed mode
3	–	Reserved

- **CLKUSABLE: UTMI Clock Usable**

0: Cleared when the UTMI 30 MHz is not usable.

1: Set when the UTMI 30 MHz is usable.

Downloaded from Arrow.com.

37.6.3 General Status Clear Register

Name: USBHS_SCR

Address: 0x40038808

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	RDERRIC	–	–	–	–

This register always reads as zero.

- **RDERRIC: Remote Device Connection Error Interrupt Clear**

0: No effect.

1: Clears the RDERRI bit in USBHS_SR.

Downloaded from Arrow.com.

37.6.4 General Status Set Register

Name: USBHS_SFR

Address: 0x4003880C

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	VBUSRQS	–
7	6	5	4	3	2	1	0
–	–	–	RDERRIS	–	–	–	–

This register always reads as zero.

- **RDERRIS: Remote Device Connection Error Interrupt Set**

0: No effect.

1: Sets the RDERRI bit in USBHS_SR, which may be useful for test or debug purposes.

- **VBUSRQS: VBUS Request Set**

Must be set to ‘1’.

Downloaded from Arrow.com.

37.6.5 Device General Control Register

Name: USBHS_DEVCTRL

Address: 0x40038000

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	OPMODE2
15	14	13	12	11	10	9	8
TSTPCKT	TSTK	TSTJ	LS	SPDCONF		RMWKUP	DETACH
7	6	5	4	3	2	1	0
ADDEN	UADD						

- **UADD: USB Address**

This field contains the device address.

This field is cleared when a USB reset is received.

- **ADDEN: Address Enable**

0: No effect.

1: Activates the UADD field (USB address).

This bit is cleared when a USB reset is received.

- **DETACH: Detach**

0: Reconnects the device.

1: Physically detaches the device (disconnects the internal pull-up resistor from D+ and D-).

- **RMWKUP: Remote Wake-Up**

0: No effect.

1: Sends an upstream resume to the host for a remote wake-up.

This bit is cleared when the USBHS receives a USB reset or once the upstream resume has been sent.

- **SPDCONF: Mode Configuration**

This field contains the peripheral speed:

Value	Name	Description
0	NORMAL	The peripheral starts in Full-speed mode and performs a high-speed reset to switch to High-speed mode if the host is high-speed-capable.
1	LOW_POWER	For a better consumption, if high speed is not needed.

Downloaded from Arrow.com.

- **LS: Low-Speed Mode Force**

0: The Full-speed mode is active.

1: The Low-speed mode is active.

This bit can be written even if USBHS_CTRL.USBE = 0 or USBHS_CTRL.FRZCLK = 1. Disabling the USBHS (by writing a zero to the USBHS_CTRL.USBE bit) does not reset this bit.

- **TSTJ: Test mode J**

0: The UTMI transceiver is in Normal operating mode.

1: The UTMI transceiver generates high-speed J state for test purposes.

- **TSTK: Test mode K**

0: The UTMI transceiver is in Normal operating mode.

1: The UTMI transceiver generates high-speed K state for test purposes.

- **TSTPCKT: Test packet mode**

0: The UTMI transceiver is in Normal operating mode.

1: The UTMI transceiver generates test packets for test purposes.

- **OPMODE2: Specific Operational mode**

0: The UTMI transceiver is in Normal operating mode.

1: The UTMI transceiver is in the “Disable bit stuffing and NRZI encoding” operational mode for test purposes.

Downloaded from Arrow.com.

37.6.6 Device Global Interrupt Status Register

Name: USBHS_DEVISR

Address: 0x40038004

Access: Read-only

31	30	29	28	27	26	25	24
DMA_7	DMA_6	DMA_5	DMA_4	DMA_3	DMA_2	DMA_1	–
23	22	21	20	19	18	17	16
–	–	PEP_9	PEP_8	PEP_7	PEP_6	PEP_5	PEP_4
15	14	13	12	11	10	9	8
PEP_3	PEP_2	PEP_1	PEP_0	–	–	–	–
7	6	5	4	3	2	1	0
–	UPRSM	EORSM	WAKEUP	EORST	SOF	MSOF	SUSP

- **SUSP: Suspend Interrupt**

0: Cleared when the USBHS_DEVICR.SUSPC bit is written to one to acknowledge the interrupt, or when the Wake-Up (WAKEUP) interrupt bit is set.

1: Set when a USB “Suspend” idle bus state has been detected for 3 frame periods (J state for 3 ms). This triggers a USB interrupt if USBHS_DEVIMR.SUSPE = 1.

- **MSOF: Micro Start of Frame Interrupt**

0: Cleared when the USBHS_DEVICR.MSOFC bit is written to one to acknowledge the interrupt.

1: Set in High-speed mode when a USB “Micro Start of Frame” PID (SOF) has been detected (every 125 µs). This triggers a USB interrupt if MSOFE = 1. The MFNUM field is updated. The FNUM field is unchanged.

- **SOF: Start of Frame Interrupt**

0: Cleared when the USBHS_DEVICR.SOFC bit is written to one to acknowledge the interrupt.

1: Set when a USB “Start of Frame” PID (SOF) has been detected (every 1 ms). This triggers a USB interrupt if SOFE = 1. The FNUM field is updated. In High-speed mode, the MFNUM field is cleared.

- **EORST: End of Reset Interrupt**

0: Cleared when the USBHS_DEVICR.EORSTC bit is written to one to acknowledge the interrupt.

1: Set when a USB “End of Reset” has been detected. This triggers a USB interrupt if USBHS_DEVIMR.EORSTE = 1.

- **WAKEUP: Wake-Up Interrupt**

0: Cleared when the USBHS_DEVICR.WAKEUPC bit is written to one to acknowledge the interrupt (USB clock inputs must be enabled before), or when the Suspend (SUSP) interrupt bit is set.

1: Set when the USBHS is reactivated by a filtered non-idle signal from the lines (not by an upstream resume). This triggers an interrupt if USBHS_DEVIMR.WAKEUPE = 1.

This interrupt is generated even if the clock is frozen by the USBHS_CTRL.FRZCLK bit.

- **EORSM: End of Resume Interrupt**

0: Cleared when the USBHS_DEVICR.EORSMC bit is written to one to acknowledge the interrupt.

1: Set when the USBHS detects a valid “End of Resume” signal initiated by the host. This triggers a USB interrupt if USBHS_DEVIMR.EORSME = 1.

Downloaded from Arrow.com.

- **UPRSM: Upstream Resume Interrupt**

0: Cleared when the USBHS_DEVICR.UPRSMC bit is written to one to acknowledge the interrupt (USB clock inputs must be enabled before).

1: Set when the USBHS sends a resume signal called “Upstream Resume”. This triggers a USB interrupt if USBHS_DEVIMR.UPRSME = 1.

- **PEP_x: Endpoint x Interrupt**

0: Cleared when the interrupt source is serviced.

1: Set when an interrupt is triggered by endpoint x (USBHS_DEVEPTISRx, USBHS_DEVEPTIMRx). This triggers a USB interrupt if USBHS_DEVIMR.PEP_x = 1.

- **DMA_x: DMA Channel x Interrupt**

0: Cleared when the USBHS_DEVDMASTATUSx interrupt source is cleared.

1: Set when an interrupt is triggered by the DMA channel x. This triggers a USB interrupt if DMA_x = 1.

Downloaded from Arrow.com.

37.6.7 Device Global Interrupt Clear Register

Name: USBHS_DEVICR

Address: 0x40038008

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	UPRSMC	EORSMC	WAKEUPC	EORSTC	SOFC	MSOFC	SUSPC

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Clears the corresponding bit in USBHS_DEVISR.

- **SUSPC: Suspend Interrupt Clear**
- **MSOFC: Micro Start of Frame Interrupt Clear**
- **SOFC: Start of Frame Interrupt Clear**
- **EORSTC: End of Reset Interrupt Clear**
- **WAKEUPC: Wake-Up Interrupt Clear**
- **EORSMC: End of Resume Interrupt Clear**
- **UPRSMC: Upstream Resume Interrupt Clear**

Downloaded from Arrow.com.

37.6.8 Device Global Interrupt Set Register

Name: USBHS_DEVIFR

Address: 0x4003800C

Access: Write-only

31	30	29	28	27	26	25	24
DMA_7	DMA_6	DMA_5	DMA_4	DMA_3	DMA_2	DMA_1	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	UPRSMS	EORSMS	WAKEUPS	EORSTS	SOFS	MSOFS	SUSPS

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Sets the corresponding bit in USBHS_DEVISR.

- **SUSPS: Suspend Interrupt Set**
- **MSOFS: Micro Start of Frame Interrupt Set**
- **SOFS: Start of Frame Interrupt Set**
- **EORSTS: End of Reset Interrupt Set**
- **WAKEUPS: Wake-Up Interrupt Set**
- **EORSMS: End of Resume Interrupt Set**
- **UPRSMS: Upstream Resume Interrupt Set**
- **DMA_x: DMA Channel x Interrupt Set**

Downloaded from Arrow.com.

37.6.9 Device Global Interrupt Mask Register

Name: USBHS_DEVIMR

Address: 0x40038010

Access: Read-only

31	30	29	28	27	26	25	24
DMA_7	DMA_6	DMA_5	DMA_4	DMA_3	DMA_2	DMA_1	–
23	22	21	20	19	18	17	16
–	–	PEP_9	PEP_8	PEP_7	PEP_6	PEP_5	PEP_4
15	14	13	12	11	10	9	8
PEP_3	PEP_2	PEP_1	PEP_0	–	–	–	–
7	6	5	4	3	2	1	0
–	UPRSME	EORSME	WAKEUPE	EORSTE	SOFE	MSOFE	SUSPE

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

- **SUSPE: Suspend Interrupt Mask**
- **MSOFE: Micro Start of Frame Interrupt Mask**
- **SOFE: Start of Frame Interrupt Mask**
- **EORSTE: End of Reset Interrupt Mask**
- **WAKEUPE: Wake-Up Interrupt Mask**
- **EORSME: End of Resume Interrupt Mask**
- **UPRSME: Upstream Resume Interrupt Mask**
- **PEP_x: Endpoint x Interrupt Mask**
- **DMA_x: DMA Channel x Interrupt Mask**

Downloaded from Arrow.com.

37.6.10 Device Global Interrupt Disable Register

Name: USBHS_DEVIDR

Address: 0x40038014

Access: Write-only

31	30	29	28	27	26	25	24
DMA_7	DMA_6	DMA_5	DMA_4	DMA_3	DMA_2	DMA_1	–
23	22	21	20	19	18	17	16
–	–	PEP_9	PEP_8	PEP_7	PEP_6	PEP_5	PEP_4
15	14	13	12	11	10	9	8
PEP_3	PEP_2	PEP_1	PEP_0	–	–	–	–
7	6	5	4	3	2	1	0
–	UPRSMEC	EORSMEC	WAKEUPEC	EORSTEC	SOFEC	MSOFEC	SUSPEC

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Clears the corresponding bit in USBHS_DEVIMR.

- **SUSPEC: Suspend Interrupt Disable**
- **MSOFEC: Micro Start of Frame Interrupt Disable**
- **SOFEC: Start of Frame Interrupt Disable**
- **EORSTEC: End of Reset Interrupt Disable**
- **WAKEUPEC: Wake-Up Interrupt Disable**
- **EORSMEC: End of Resume Interrupt Disable**
- **UPRSMEC: Upstream Resume Interrupt Disable**
- **PEP_x: Endpoint x Interrupt Disable**
- **DMA_x: DMA Channel x Interrupt Disable**

Downloaded from Arrow.com.

37.6.11 Device Global Interrupt Enable Register

Name: USBHS_DEVIER

Address: 0x40038018

Access: Write-only

31	30	29	28	27	26	25	24
DMA_7	DMA_6	DMA_5	DMA_4	DMA_3	DMA_2	DMA_1	–
23	22	21	20	19	18	17	16
–	–	PEP_9	PEP_8	PEP_7	PEP_6	PEP_5	PEP_4
15	14	13	12	11	10	9	8
PEP_3	PEP_2	PEP_1	PEP_0	–	–	–	–
7	6	5	4	3	2	1	0
–	UPRSMES	EORSMES	WAKEUPES	EORSTES	SOFES	MSOFES	SUSPES

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Sets the corresponding bit in USBHS_DEVIMR.

- **SUSPES: Suspend Interrupt Enable**
- **MSOFES: Micro Start of Frame Interrupt Enable**
- **SOFES: Start of Frame Interrupt Enable**
- **EORSTES: End of Reset Interrupt Enable**
- **WAKEUPES: Wake-Up Interrupt Enable**
- **EORSMES: End of Resume Interrupt Enable**
- **UPRSMES: Upstream Resume Interrupt Enable**
- **PEP_x: Endpoint x Interrupt Enable**
- **DMA_x: DMA Channel x Interrupt Enable**

Downloaded from Arrow.com.

37.6.12 Device Endpoint Register

Name: USBHS_DEVEPT

Address: 0x4003801C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	EPRST9	EPRST8
23	22	21	20	19	18	17	16
EPRST7	EPRST6	EPRST5	EPRST4	EPRST3	EPRST2	EPRST1	EPRST0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	EPEN9	EPEN8
7	6	5	4	3	2	1	0
EPEN7	EPEN6	EPEN5	EPEN4	EPEN3	EPEN2	EPEN1	EPEN0

- **EPENx: Endpoint x Enable**

0: Endpoint x is disabled, forcing the endpoint x state to inactive (no answer to USB requests) and resetting the endpoint x registers (USBHS_DEVEPTCFGx, USBHS_DEVEPTISRx, USBHS_DEVEPTIMRx) but not the endpoint configuration (USBHS_DEVEPTCFGx.ALLOC, USBHS_DEVEPTCFGx.EPBK, USBHS_DEVEPTCFGx.EPSIZE, USBHS_DEVEPTCFGx.EPDIR, USBHS_DEVEPTCFGx.EPTYPE).

1: Endpoint x is enabled.

- **EPRSTx: Endpoint x Reset**

0: Completes the reset operation and starts using the FIFO.

1: Resets the endpoint x FIFO prior to any other operation, upon hardware reset or when a USB bus reset has been received. This resets the endpoint x registers (USBHS_DEVEPTCFGx, USBHS_DEVEPTISRx, USBHS_DEVEPTIMRx) but not the endpoint configuration (USBHS_DEVEPTCFGx.ALLOC, USBHS_DEVEPTCFGx.EPBK, USBHS_DEVEPTCFGx.EPSIZE, USBHS_DEVEPTCFGx.EPDIR, USBHS_DEVEPTCFGx.EPTYPE).

The whole endpoint mechanism (FIFO counter, reception, transmission, etc.) is reset apart from the Data Toggle Sequence field (USBHS_DEVEPTISRx.DTSEQ), which can be cleared by setting the USBHS_DEVEPTIMRx.RSTDT bit (by writing a one to the USBHS_DEVEPTIERx.RSTDTS bit).

The endpoint configuration remains active and the endpoint is still enabled.

This bit is cleared upon receiving a USB reset.

Downloaded from Arrow.com.

37.6.13 Device Frame Number Register

Name: USBHS_DEVFNUM

Address: 0x40038020

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
FNCERR	–	FNUM					
7	6	5	4	3	2	1	0
FNUM					MFNUM		

- **MFNUM: Micro Frame Number**

This field contains the 3-bit micro frame number information. It is provided in the last received MSOF packet.

This field is cleared at the beginning of each start of frame (SOF interrupt) or upon receiving a USB reset.

MFNUM is updated even if a corrupted MSOF is received.

- **FNUM: Frame Number**

This field contains the 11-bit frame number information. It is provided in the last received SOF packet.

This field is cleared upon receiving a USB reset.

FNUM is updated even if a corrupted SOF is received.

- **FNCERR: Frame Number CRC Error**

0: Cleared upon receiving a USB reset.

1: Set when a corrupted frame number (or microframe number) is received. This bit and the SOF (or MSOF) interrupt bit are updated at the same time.

Downloaded from Arrow.com.

37.6.14 Device Endpoint x Configuration Register

Name: USBHS_DEVEPTCFGx [x=0..9]

Address: 0x40038100

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	NBTRANS		EPTYPE		–	AUTOSW	EPDIR
7	6	5	4	3	2	1	0
–	EPSIZE			EPBK		ALLOC	–

- **ALLOC: Endpoint Memory Allocate**

0: Frees the endpoint memory.

1: Allocates the endpoint memory. The user should check the USBHS_DEVEPTISRx.CFGOK bit to know whether the allocation of this endpoint is correct.

This bit is cleared upon receiving a USB reset (except for endpoint 0).

- **EPBK: Endpoint Banks**

This field should be written to select the number of banks for the endpoint:

Value	Name	Description
0	1_BANK	Single-bank endpoint
1	2_BANK	Double-bank endpoint
2	3_BANK	Triple-bank endpoint
3	–	Reserved

For control endpoints, a single-bank endpoint (0b00) should be selected.

This field is cleared upon receiving a USB reset (except for endpoint 0).

- **EPSIZE: Endpoint Size**

This field should be written to select the size of each endpoint bank:

Value	Name	Description
0	8_BYTE	8 bytes
1	16_BYTE	16 bytes
2	32_BYTE	32 bytes
3	64_BYTE	64 bytes
4	128_BYTE	128 bytes
5	256_BYTE	256 bytes
6	512_BYTE	512 bytes
7	1024_BYTE	1024 bytes

This field is cleared upon receiving a USB reset (except for endpoint 0).

Downloaded from Arrow.com.

- **EPDIR: Endpoint Direction**

This bit is cleared upon receiving a USB reset.

0 (OUT): The endpoint direction is OUT.

1 (IN): The endpoint direction is IN (nor for control endpoints).

- **AUTOSW: Automatic Switch**

This bit is cleared upon receiving a USB reset.

0: The automatic bank switching is disabled.

1: The automatic bank switching is enabled.

- **EPTYPE: Endpoint Type**

This field should be written to select the endpoint type:

Value	Name	Description
0	CTRL	Control
1	ISO	Isochronous
2	BLK	Bulk
3	INTRPT	Interrupt

This field is cleared upon receiving a USB reset.

- **NBTRANS: Number of transactions per microframe for isochronous endpoint**

This field should be written with the number of transactions per microframe to perform high-bandwidth isochronous transfer.

It can be written only for endpoints that have this capability (see USBHS_FEATURES.ENHBISOx bit). Otherwise, this field is 0.

This field is irrelevant for non-isochronous endpoints.

Value	Name	Description
0	0_TRANS	Reserved to endpoint that does not have the high-bandwidth isochronous capability.
1	1_TRANS	Default value: one transaction per microframe.
2	2_TRANS	Two transactions per microframe. This endpoint should be configured as double-bank.
3	3_TRANS	Three transactions per microframe. This endpoint should be configured as triple-bank.

Downloaded from Arrow.com.

37.6.15 Device Endpoint x Status Register (Control, Bulk, Interrupt Endpoints)

Name: USBHS_DEVEPTISRx [x=0..9]

Address: 0x40038130

Access: Read-only 0x0100

31	30	29	28	27	26	25	24
–	BYCT						
23	22	21	20	19	18	17	16
BYCT				–	CFGOK	CTRLDIR	RWALL
15	14	13	12	11	10	9	8
CURRBK		NBUSYBK		–	–	DTSEQ	
7	6	5	4	3	2	1	0
SHORTPACKET	STALLEDI	OVERFI	NAKINI	NAKOUTI	RXSTPI	RXOUTI	TXINI

This register view is relevant only if EPTYPE = 0x0, 0x2or 0x3 in “Device Endpoint x Configuration Register” on page 616.

- **TXINI: Transmitted IN Data Interrupt**

For control endpoints:

0: Cleared when TXINIC = 1. This acknowledges the interrupt and sends the packet.

1: Set when the current bank is ready to accept a new IN packet. This triggers a PEP_x interrupt if TXINE = 1.

For bulk and interrupt IN endpoints:

0: Cleared when TXINIC = 1. This acknowledges the interrupt, which has no effect on the endpoint FIFO. USBHS_DEVEPTISRx.TXINI shall always be cleared before clearing USBHS_DEVEPTIMRx.FIFOCON.

1: Set at the same time as USBHS_DEVEPTIMRx.FIFOCON when the current bank is free. This triggers a PEP_x interrupt if TXINE = 1.

The user writes into the FIFO and clears the USBHS_DEVEPTIMRx.FIFOCON bit to allow the USBHS to send the data. If the IN endpoint is composed of multiple banks, this also switches to the next bank. The USBHS_DEVEPTISRx.TXINI and USBHS_DEVEPTIMRx.FIFOCON bits are set/cleared in accordance with the status of the next bank.

This bit is inactive (cleared) for bulk and interrupt OUT endpoints.

- **RXOUTI: Received OUT Data Interrupt**

For control endpoints:

0: Cleared by writing a one to the RXOUTIC bit. This acknowledges the interrupt and frees the bank.

1: Set when the current bank contains a bulk OUT packet (data or status stage). This triggers a PEP_x interrupt if USBHS_DEVEPTIMRx.RXOUTE = 1.

For bulk and interrupt OUT endpoints:

0: Cleared by writing a one to the RXOUTIC bit. This acknowledges the interrupt, which has no effect on the endpoint FIFO. USBHS_DEVEPTISRx.RXOUTI shall always be cleared before clearing USBHS_DEVEPTIMRx.FIFOCON.

1: Set at the same time as USBHS_DEVEPTIMRx.FIFOCON when the current bank is full. This triggers a PEP_x interrupt if USBHS_DEVEPTIMRx.RXOUTE = 1.

The user reads from the FIFO and clears the USBHS_DEVEPTIMRx.FIFOCON bit to free the bank. If the OUT endpoint is composed of multiple banks, this also switches to the next bank. The USBHS_DEVEPTISRx.RXOUTI and USBHS_DEVEPTIMRx.FIFOCON bits are set/cleared in accordance with the status of the next bank.

This bit is inactive (cleared) for bulk and interrupt IN endpoints.

Downloaded from Arrow.com.

- **RXSTPI: Received SETUP Interrupt**

This bit is set, for control endpoints, to signal that the current bank contains a new valid SETUP packet. This triggers a PEP_x interrupt if RXSTPE = 1.

It is cleared by writing a one to the RXSTPIC bit. This acknowledges the interrupt and frees the bank.

This bit is inactive (cleared) for bulk and interrupt IN/OUT endpoints.

- **NAKOUTI: NAKed OUT Interrupt**

0: Cleared when NAKOUTIC = 1. This acknowledges the interrupt.

1: Set when a NAK handshake has been sent in response to an OUT request from the host. This triggers a PEP_x interrupt if NAKOUTE = 1.

- **NAKINI: NAKed IN Interrupt**

0: Cleared when NAKINIC = 1. This acknowledges the interrupt.

1: Set when a NAK handshake has been sent in response to an IN request from the host. This triggers a PEP_x interrupt if NAKINE = 1.

- **OVERFI: Overflow Interrupt**

0: Cleared when the OVERFIC bit is written to one. This acknowledges the interrupt.

1: Set when an overflow error occurs. This triggers a PEP_x interrupt if OVERFE = 1.

For all endpoint types, an overflow can occur during the OUT stage if the host attempts to write into a bank that is too small for the packet. The packet is acknowledged and the USBHS_DEVEPTISRx.RXOUTI bit is set as if no overflow had occurred. The bank is filled with all the first bytes of the packet that fit in.

- **STALLEDI: STALLed Interrupt**

0: Cleared when STALLEDIC = 1. This acknowledges the interrupt.

1: Set to signal that a STALL handshake has been sent. To do that, the software has to set the STALLRQ bit (by writing a one to the STALLRQS bit). This triggers a PEP_x interrupt if STALLEDE = 1.

- **SHORTPACKET: Short Packet Interrupt**

0: Cleared when SHORTPACKETC = 1. This acknowledges the interrupt.

1: Set for non-control OUT endpoints, when a short packet has been received. This triggers a PEP_x interrupt if USBHS_DEVEPTIMRx.SHORTPACKETE = 1.

- **DTSEQ: Data Toggle Sequence**

This field is set to indicate the PID of the current bank:

Value	Name	Description
0	DATA0	Data0 toggle sequence
1	DATA1	Data1 toggle sequence
2	DATA2	Reserved for high-bandwidth isochronous endpoint
3	MDATA	Reserved for high-bandwidth isochronous endpoint

For IN transfers, it indicates the data toggle sequence that should be used for the next packet to be sent. This is not relative to the current bank.

For OUT transfers, this value indicates the last data toggle sequence received on the current bank.

Downloaded from Arrow.com.

By default, DTSEQ is 0b01, as if the last data toggle sequence was Data1, so the next sent or expected data toggle sequence should be Data0.

- **NBUSYBK: Number of Busy Banks**

This field is set to indicate the number of busy banks:

Value	Name	Description
0	0_BUSY	0 busy bank (all banks free)
1	1_BUSY	1 busy bank
2	2_BUSY	2 busy banks
3	3_BUSY	3 busy banks

For IN endpoints, it indicates the number of banks filled by the user and ready for IN transfer. When all banks are free, this triggers a PEP_x interrupt if NBUSYBKE = 1.

For OUT endpoints, it indicates the number of banks filled by OUT transactions from the host. When all banks are busy, this triggers a PEP_x interrupt if NBUSYBKE = 1.

When the USBHS_DEVEPTIMRx.FIFOCON bit is cleared (by writing a one to the USBHS_DEVEPTIMRx.FIFOCONC bit) to validate a new bank, this field is updated two or three clock cycles later to calculate the address of the next bank.

A PEP_x interrupt is triggered if:

- for IN endpoint, USBHS_DEVEPTIMRx.NBUSYBKE = 1 and all the banks are free;
- for OUT endpoint, USBHS_DEVEPTIMRx.NBUSYBKE = 1 and all the banks are busy.

- **CURRBK: Current Bank**

This bit is set for non-control endpoints, to indicate the current bank:

Value	Name	Description
0	BANK0	Current bank is bank0
1	BANK1	Current bank is bank1
2	BANK2	Current bank is bank2
3	–	Reserved

This field may be updated one clock cycle after the RWALL bit changes, so the user should not poll this field as an interrupt bit.

- **RWALL: Read/Write Allowed**

This bit is set for IN endpoints when the current bank is not full, i.e., the user can write further data into the FIFO.

This bit is set for OUT endpoints when the current bank is not empty, i.e., the user can read further data from the FIFO.

This bit is never set if USBHS_DEVEPTIMRx.STALLRQ = 1 or in case of error.

This bit is cleared otherwise.

This bit should not be used for control endpoints.

- **CTRLDIR: Control Direction**

0: Cleared after a SETUP packet to indicate that the following packet is an OUT packet.

1: Set after a SETUP packet to indicate that the following packet is an IN packet.

Downloaded from Arrow.com.

- **CFGOK: Configuration OK Status**

This bit is updated when USBHS_DEVEPTCFGx.ALLOC = 1.

This bit is set if the endpoint x number of banks (USBHS_DEVEPTCFGx.EPBK) and size (USBHS_DEVEPTCFGx.EPSIZE) are correct compared to the maximal allowed number of banks and size for this endpoint and to the maximal FIFO size (i.e., the DPRAM size).

If this bit is cleared, the user should rewrite correct values to the USBHS_DEVEPTCFGx.EPBK and USBHS_DEVEPTCFGx.EPSIZE fields.

- **BYCT: Byte Count**

This field is set with the byte count of the FIFO.

For IN endpoints, the field is incremented after each byte written by the software into the endpoint and decremented after each byte sent to the host.

For OUT endpoints, the field is incremented after each byte received from the host and decremented after each byte read by the software from the endpoint.

This field may be updated one clock cycle after the RWALL bit changes, so the user should not poll this field as an interrupt bit.

Downloaded from Arrow.com.

37.6.16 Device Endpoint x Status Register (Isochronous Endpoints)

Name: USBHS_DEVEPTISRx [x=0..9] (ISOENPT)

Address: 0x40038130

Access: Read-only 0x0100

31	30	29	28	27	26	25	24
–	BYCT						
23	22	21	20	19	18	17	16
BYCT				–	CFGOK	–	RWALL
15	14	13	12	11	10	9	8
CURRBK		NBUSYBK		–	ERRORTRANS	DTSEQ	
7	6	5	4	3	2	1	0
SHORTPACKET	CRCERRI	OVERFI	HBISOFLUSHI	HBISOINERRI	UNDERFI	RXOUTI	TXINI

This register view is relevant only if EPTYPE = 0x1 in “Device Endpoint x Configuration Register” on page 616.

- **TXINI: Transmitted IN Data Interrupt**

For control endpoints:

0: Cleared when TXINIC = 1. This acknowledges the interrupt and sends the packet.

1: Set when the current bank is ready to accept a new IN packet. This triggers a PEP_x interrupt if TXINE = 1.

For IN endpoints:

0: Cleared when TXINIC = 1. This acknowledges the interrupt, which has no effect on the endpoint FIFO. USBHS_DEVEPTISRx.TXINI shall always be cleared before clearing USBHS_DEVEPTIMRx.FIFOCON.

1: Set at the same time as USBHS_DEVEPTIMRx.FIFOCON when the current bank is free. This triggers a PEP_x interrupt if TXINE = 1.

The user writes into the FIFO and clears the USBHS_DEVEPTIMRx.FIFOCON bit to allow the USBHS to send the data. If the IN endpoint is composed of multiple banks, this also switches to the next bank. The USBHS_DEVEPTISRx.TXINI and USBHS_DEVEPTIMRx.FIFOCON bits are set/cleared in accordance with the status of the next bank.

This bit is inactive (cleared) for OUT endpoints.

- **RXOUTI: Received OUT Data Interrupt**

For control endpoints:

0: Cleared by writing a one to the RXOUTIC bit. This acknowledges the interrupt and frees the bank.

1: Set when the current bank contains a bulk OUT packet (data or status stage). This triggers a PEP_x interrupt if USBHS_DEVEPTIMRx.RXOUTE = 1.

For OUT endpoints:

0: Cleared by writing a one to the RXOUTIC bit. This acknowledges the interrupt, which has no effect on the endpoint FIFO. USBHS_DEVEPTISRx.RXOUTI shall always be cleared before clearing USBHS_DEVEPTIMRx.FIFOCON.

1: Set at the same time as USBHS_DEVEPTIMRx.FIFOCON when the current bank is full. This triggers a PEP_x interrupt if USBHS_DEVEPTIMRx.RXOUTE = 1.

The user reads from the FIFO and clears the USBHS_DEVEPTIMRx.FIFOCON bit to free the bank. If the OUT endpoint is composed of multiple banks, this also switches to the next bank. The USBHS_DEVEPTISRx.RXOUTI and USBHS_DEVEPTIMRx.FIFOCON bits are set/cleared in accordance with the status of the next bank.

This bit is inactive (cleared) for IN endpoints.

Downloaded from Arrow.com.

- **UNDERFI: Underflow Interrupt**

This bit is set, for isochronous IN/OUT endpoints, when an underflow error occurs. This triggers a PEP_x interrupt if UNDERFE = 1.

An underflow can occur during IN stage if the host attempts to read from an empty bank. A zero-length packet is then automatically sent by the USBHS.

An underflow can also occur during OUT stage if the host sends a packet while the bank is already full. Typically, the CPU is not fast enough. The packet is lost.

It is cleared by writing a one to the UNDERFIC bit. This acknowledges the interrupt.

- **HBISOINERRI: High Bandwidth Isochronous IN Underflow Error Interrupt**

0: Cleared when the HBISOINERRIC bit is written to one. This acknowledges the interrupt.

1: Set for High-bandwidth isochronous IN endpoint (with NBTRANS = 2 or 3) at the end of the microframe, if less than N banks were written by the CPU within this microframe. This triggers a PEP_x interrupt if HBISOINERRE = 1.

- **HBISOFLUSHI: High Bandwidth Isochronous IN Flush Interrupt**

0: Cleared when the HBISOFLUSHIC bit is written to one. This acknowledges the interrupt.

1: Set for High-bandwidth isochronous IN endpoint (with NBTRANS = 2 or 3) at the end of the microframe, if less than N transactions have been completed by the USBHS without underflow error. This may occur in case of a missing IN token. In this case, the banks are flushed out to ensure the data synchronization between the host and the device. This triggers a PEP_x interrupt if HBISOFLUSHE = 1.

- **OVERFI: Overflow Interrupt**

0: Cleared when OVERFIC = 1. This acknowledges the interrupt.

1: Set when an overflow error occurs. This triggers a PEP_x interrupt if OVERFE = 1. For all endpoint types, an overflow can occur during OUT stage if the host attempts to write into a bank that is too small for the packet. The packet is acknowledged and the USBHS_DEVEPTISRx.RXOUTI bit is set as if no overflow had occurred. The bank is filled with all the first bytes of the packet that fit in.

- **CRCERRI: CRC Error Interrupt**

0: Cleared when CRCERRIC = 1. This acknowledges the interrupt.

1: Set to signal that a CRC error has been detected in an isochronous OUT endpoint. The OUT packet is stored in the bank as if no CRC error had occurred. This triggers a PEP_x interrupt if CRCERRE = 1.

- **SHORTPACKET: Short Packet Interrupt**

0: Cleared when SHORTPACKETC = 1. This acknowledges the interrupt.

1: Set for non-control OUT endpoints, when a short packet has been received. This triggers a PEP_x interrupt if USBHS_DEVEPTIMRx.SHORTPACKETE = 1.

Downloaded from Arrow.com.

- **DTSEQ: Data Toggle Sequence**

This field is set to indicate the PID of the current bank:

Value	Name	Description
0	DATA0	Data0 toggle sequence
1	DATA1	Data1 toggle sequence
2	DATA2	Data2 toggle sequence (for high-bandwidth isochronous endpoint)
3	MDATA	MData toggle sequence (for high-bandwidth isochronous endpoint)

For IN transfers, it indicates the data toggle sequence that should be used for the next packet to be sent. This is not relative to the current bank.

For OUT transfers, this value indicates the last data toggle sequence received on the current bank.

By default, DTSEQ is 0b01, as if the last data toggle sequence was Data1, so the next sent or expected data toggle sequence should be Data0.

For high-bandwidth isochronous endpoint, a PEP_x interrupt is triggered if:

- USBHS_DEVEPTIMRx.MDATAE = 1 and a MData packet has been received (DTSEQ = MData and USBHS_DEVEPTISRx.RXOUTI = 1).
- USBHS_DEVEPTISRx.DATAXE = 1 and a Data0/1/2 packet has been received (DTSEQ = Data0/1/2 and USBHS_DEVEPTISRx.RXOUTI = 1).

- **ERRORTRANS: High-bandwidth Isochronous OUT Endpoint Transaction Error Interrupt**

This bit is set when a transaction error occurs during the current microframe (the data toggle sequencing is not compliant with the USB 2.0 standard). This triggers a PEP_x interrupt if USBHS_DEVEPTIMRx.ERRORTRANSE = 1.

This bit is set as long as the current bank (CURRBK) belongs to the bad n-transactions (n = 1, 2 or 3) transferred during the microframe. It is cleared by software by clearing (at least once) the USBHS_DEVEPTIMRx.FIFOCON bit to switch to the bank that belongs to the next n-transactions (next microframe).

- **NBUSYBK: Number of Busy Banks**

This field is set to indicate the number of busy banks:

Value	Name	Description
0	0_BUSY	0 busy bank (all banks free)
1	1_BUSY	1 busy bank
2	2_BUSY	2 busy banks
3	3_BUSY	3 busy banks

For IN endpoints, it indicates the number of banks filled by the user and ready for IN transfer. When all banks are free, this triggers a PEP_x interrupt if NBUSYBKE = 1.

For OUT endpoints, it indicates the number of banks filled by OUT transactions from the host. When all banks are busy, this triggers a PEP_x interrupt if NBUSYBKE = 1.

When the USBHS_DEVEPTIMRx.FIFOCON bit is cleared (by writing a one to the USBHS_DEVEPTIMRx.FIFOCONC bit) to validate a new bank, this field is updated two or three clock cycles later to calculate the address of the next bank.

A PEP_x interrupt is triggered if:

- For IN endpoint, USBHS_DEVEPTIMRx.NBUSYBKE = 1 and all the banks are free.
- For OUT endpoint, USBHS_DEVEPTIMRx.NBUSYBKE = 1 and all the banks are busy.

Downloaded from Arrow.com.

- **CURRBK: Current Bank**

This field is used to indicate the current bank. It may be updated one clock cycle after the RWALL bit changes, so the user should not poll this field as an interrupt bit.

Value	Name	Description
0	BANK0	Current bank is bank0
1	BANK1	Current bank is bank1
2	BANK2	Current bank is bank2
3	–	Reserved

- **RWALL: Read/Write Allowed**

This bit is set for IN endpoints when the current bank is not full, i.e., the user can write further data into the FIFO.

This bit is set for OUT endpoints when the current bank is not empty, i.e., the user can read further data from the FIFO.

This bit is never set in case of error.

This bit is cleared otherwise.

- **CFGOK: Configuration OK Status**

This bit is updated when USBHS_DEVEPTCFGx.ALLOC = 1.

This bit is set if the endpoint x number of banks (USBHS_DEVEPTCFGx.EPBK) and size (USBHS_DEVEPTCFGx.EPSIZE) are correct compared to the maximal allowed number of banks and size for this endpoint and to the maximal FIFO size (i.e., the DPRAM size).

If this bit is cleared, the user should rewrite correct values to the USBHS_DEVEPTCFGx.EPBK and USBHS_DEVEPTCFGx.EPSIZE fields.

- **BYCT: Byte Count**

This field is set with the byte count of the FIFO.

For IN endpoints, the field is incremented after each byte written by the software into the endpoint and decremented after each byte sent to the host.

For OUT endpoints, the field is incremented after each byte received from the host and decremented after each byte read by the software from the endpoint.

This field may be updated one clock cycle after the RWALL bit changes, so the user should not poll this field as an interrupt bit.

Downloaded from Arrow.com.

37.6.17 Device Endpoint x Clear Register (Control, Bulk, Interrupt Endpoints)

Name: USBHS_DEVEPTICRx [x=0..9]

Address: 0x40038160

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
SHORTPACKETC	STALLEDIC	OVERFIC	NAKINIC	NAKOUTIC	RXSTPIC	RXOUTIC	TXINIC

This register view is relevant only if EPTYPE = 0x0, 0x2 or 0x3 in “Device Endpoint x Configuration Register” on page 616.

For additional information, see “Device Endpoint x Status Register (Control, Bulk, Interrupt Endpoints)” on page 618.

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Clears the corresponding bit in USBHS_DEVEPTISRx.

- **TXINIC: Transmitted IN Data Interrupt Clear**
- **RXOUTIC: Received OUT Data Interrupt Clear**
- **RXSTPIC: Received SETUP Interrupt Clear**
- **NAKOUTIC: NAKed OUT Interrupt Clear**
- **NAKINIC: NAKed IN Interrupt Clear**
- **OVERFIC: Overflow Interrupt Clear**
- **STALLEDIC: STALLed Interrupt Clear**
- **SHORTPACKETC: Short Packet Interrupt Clear**

Downloaded from Arrow.com.

37.6.18 Device Endpoint x Clear Register (Isochronous Endpoints)

Name: USBHS_DEVEPTICRx [x=0..9] (ISOENPT)

Address: 0x40038160

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
SHORTPACKETC	CRCERRIC	OVERFIC	HBISOFLUSHIC	HBISOINERRIC	UNDERFIC	RXOUTIC	TXINIC

This register view is relevant only if EPTYPE = 0x1 in “Device Endpoint x Configuration Register” on page 616.

For additional information, see “Device Endpoint x Status Register (Isochronous Endpoints)” on page 622.

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Clears the corresponding bit in USBHS_DEVEPTISRx.

- **TXINIC: Transmitted IN Data Interrupt Clear**
- **RXOUTIC: Received OUT Data Interrupt Clear**
- **UNDERFIC: Underflow Interrupt Clear**
- **HBISOINERRIC: High Bandwidth Isochronous IN Underflow Error Interrupt Clear**
- **HBISOFLUSHIC: High Bandwidth Isochronous IN Flush Interrupt Clear**
- **OVERFIC: Overflow Interrupt Clear**
- **CRCERRIC: CRC Error Interrupt Clear**
- **SHORTPACKETC: Short Packet Interrupt Clear**

Downloaded from Arrow.com.

37.6.19 Device Endpoint x Set Register (Control, Bulk, Interrupt Endpoints)

Name: USBHS_DEVEPTIFRx [x=0..9]

Address: 0x40038190

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	NBUSYBKS	–	–	–	–
7	6	5	4	3	2	1	0
SHORTPACKETS	STALLEDIS	OVERFIS	NAKINIS	NAKOUTIS	RXSTPIS	RXOUTIS	TXINIS

This register view is relevant only if EPTYPE = 0x0, 0x2 or 0x3 in “Device Endpoint x Configuration Register” on page 616.

For additional information, see “Device Endpoint x Status Register (Control, Bulk, Interrupt Endpoints)” on page 618.This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Sets the corresponding bit in USBHS_DEVEPTISRx, which may be useful for test or debug purposes.

- **TXINIS: Transmitted IN Data Interrupt Set**
- **RXOUTIS: Received OUT Data Interrupt Set**
- **RXSTPIS: Received SETUP Interrupt Set**
- **NAKOUTIS: NAKed OUT Interrupt Set**
- **NAKINIS: NAKed IN Interrupt Set**
- **OVERFIS: Overflow Interrupt Set**
- **STALLEDIS: STALLed Interrupt Set**
- **SHORTPACKETS: Short Packet Interrupt Set**
- **NBUSYBKS: Number of Busy Banks Interrupt Set**

Downloaded from Arrow.com.

37.6.20 Device Endpoint x Set Register (Isochronous Endpoints)

Name: USBHS_DEVEPTIFRx [x=0..9] (ISOENPT)

Address: 0x40038190

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	NBUSYBKS	–	–	–	–
7	6	5	4	3	2	1	0
SHORTPACKETS	CRCERRIS	OVERFIS	HBISOFLUSHIS	HBISOINERRIS	UNDERFIS	RXOUTIS	TXINIS

This register view is relevant only if EPTYPE = 0x1 in “Device Endpoint x Configuration Register” on page 616.

For additional information, see “Device Endpoint x Status Register (Isochronous Endpoints)” on page 622.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Sets the corresponding bit in USBHS_DEVEPTISRx, which may be useful for test or debug purposes.

- **TXINIS: Transmitted IN Data Interrupt Set**
- **RXOUTIS: Received OUT Data Interrupt Set**
- **UNDERFIS: Underflow Interrupt Set**
- **HBISOINERRIS: High Bandwidth Isochronous IN Underflow Error Interrupt Set**
- **HBISOFLUSHIS: High Bandwidth Isochronous IN Flush Interrupt Set**
- **OVERFIS: Overflow Interrupt Set**
- **CRCERRIS: CRC Error Interrupt Set**
- **SHORTPACKETS: Short Packet Interrupt Set**
- **NBUSYBKS: Number of Busy Banks Interrupt Set**

Downloaded from Arrow.com.

37.6.21 Device Endpoint x Mask Register (Control, Bulk, Interrupt Endpoints)

Name: USBHS_DEVEPTIMRx [x=0..9]

Address: 0x400381C0

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	STALLRQ	RSTDT	NYETDIS	EPDISHDMA
15	14	13	12	11	10	9	8
–	FIFOCON	KILLBK	NBUSYBKE	–	–	–	–
7	6	5	4	3	2	1	0
SHORTPACKETE	STALLEDE	OVERFE	NAKINE	NAKOUTE	RXSTPE	RXOUTE	TXINE

This register view is relevant only if EPTYPE = 0x0, 0x2 or 0x3 in “Device Endpoint x Configuration Register” on page 616.

- **TXINE: Transmitted IN Data Interrupt**

0: Cleared when USBHS_DEVEPTIDRx.TXINEC = 1. This disables the Transmitted IN Data interrupt (USBHS_DEVEPTISRx.TXINI).

1: Set when USBHS_DEVEPTIERx.TXINES = 1. This enables the Transmitted IN Data interrupt (USBHS_DEVEPTISRx.TXINI).

- **RXOUTE: Received OUT Data Interrupt**

0: Cleared when USBHS_DEVEPTIDRx.RXOUTEC = 1. This disables the Received OUT Data interrupt (USBHS_DEVEPTISRx.RXOUTI).

1: Set when USBHS_DEVEPTIERx.RXOUTES = 1. This enables the Received OUT Data interrupt (USBHS_DEVEPTISRx.RXOUTI).

- **RXSTPE: Received SETUP Interrupt**

0: Cleared when USBHS_DEVEPTIERx.RXSTPEC = 1. This disables the Received SETUP interrupt (USBHS_DEVEPTISRx.RXSTPI).

1: Set when USBHS_DEVEPTIERx.RXSTPES = 1. This enables the Received SETUP interrupt (USBHS_DEVEPTISRx.RXSTPI).

- **NAKOUTE: NAKed OUT Interrupt**

0: Cleared when USBHS_DEVEPTIDRx.NAKOUTEC = 1. This disables the NAKed OUT interrupt (USBHS_DEVEPTISRx.NAKOUTI).

1: Set when USBHS_DEVEPTIERx.NAKOUTES = 1. This enables the NAKed OUT interrupt (USBHS_DEVEPTISRx.NAKOUTI).

- **NAKINE: NAKed IN Interrupt**

0: Cleared when USBHS_DEVEPTIDRx.NAKINEC = 1. This disables the NAKed IN interrupt (USBHS_DEVEPTISRx.NAKINI).

1: Set when USBHS_DEVEPTIERx.NAKINES = 1. This enables the NAKed IN interrupt (USBHS_DEVEPTISRx.NAKINI).

Downloaded from Arrow.com.

- **OVERFE: Overflow Interrupt**

0: Cleared when USBHS_DEVEPTIDRx.OVERFEC = 1. This disables the Overflow interrupt (USBHS_DEVEPTISRx.OVERFI).

1: Set when USBHS_DEVEPTIERx.OVERFES = 1. This enables the Overflow interrupt (USBHS_DEVEPTISRx.OVERFI).

- **STALLEDE: STALLed Interrupt**

0: Cleared when USBHS_DEVEPTIDRx.STALLEDEC = 1. This disables the STALLed interrupt (USBHS_DEVEPTISRx.STALLEDI).

1: Set when USBHS_DEVEPTIERx.STALLEDES = 1. This enables the STALLed interrupt (USBHS_DEVEPTISRx.STALLEDI).

- **SHORTPACKETE: Short Packet Interrupt**

0: Cleared when USBHS_DEVEPTIDRx.SHORTPACKETEC = 1. This disables the Short Packet interrupt (USBHS_DEVEPTISRx.SHORTPACKET).

1: Set when USBHS_DEVEPTIERx.SHORTPACKETES = 1. This enables the Short Packet interrupt (USBHS_DEVEPTISRx.SHORTPACKET).

If this bit is set for non-control IN endpoints, a short packet transmission is guaranteed upon ending a DMA transfer, thus signaling an end of isochronous frame or a bulk or interrupt end of transfer, provided that the End of DMA Buffer Output Enable (END_B_EN) bit and the Automatic Switch (AUTOSW) = 1.

- **NBUSYBKE: Number of Busy Banks Interrupt**

0: Cleared when USBHS_DEVEPTIDRx.NBUSYBKEC = 0. This disables the Number of Busy Banks interrupt (USBHS_DEVEPTISRx.NBUSYBK).

1: Set when the USBHS_DEVEPTIERx.NBUSYBKES = 1. This enables the Number of Busy Banks interrupt (USBHS_DEVEPTISRx.NBUSYBK).

- **KILLBK: Kill IN Bank**

This bit is set when the USBHS_DEVEPTIERx.KILLBKS bit is written to one. This kills the last written bank.

This bit is cleared when the bank is killed.

CAUTION: The bank is really cleared when the “kill packet” procedure is accepted by the USBHS core. This bit is automatically cleared after the end of the procedure:

The bank is really killed: USBHS_DEVEPTISRx.NBUSYBK is decremented.

The bank is not cleared but sent (IN transfer): USBHS_DEVEPTISRx.NBUSYBK is decremented.

The bank is not cleared because it was empty.

The user should wait for this bit to be cleared before trying to kill another packet.

This kill request is refused if at the same time an IN token is coming and the last bank is the current one being sent on the USB line. If at least two banks are ready to be sent, there is no problem to kill a packet even if an IN token is coming. Indeed, in this case, the current bank is sent (IN transfer) while the last bank is killed.

Downloaded from Arrow.com.

- **FIFOCON: FIFO Control**

For control endpoints:

The FIFOCON and RWALL bits are irrelevant. Therefore, the software never uses them on these endpoints. When read, their value is always 0.

For IN endpoints:

0: Cleared (by writing a one to the USBHS_DEVEPTIDRx.FIFOCONC bit) to send the FIFO data and to switch to the next bank.

1: Set when the current bank is free, at the same time as USBHS_DEVEPTISRx.TXINI.

For OUT endpoints:

0: Cleared (by writing a one to the USBHS_DEVEPTIDRx.FIFOCONC bit) to free the current bank and to switch to the next bank.

1: Set when the current bank is full, at the same time as USBHS_DEVEPTISRx.RXOUTI.

- **EPDISHDMA: Endpoint Interrupts Disable HDMA Request**

This bit is set when USBHS_DEVEPTIERx.EPDISHDMAS = 1. This pauses the on-going DMA channel x transfer on any Endpoint x interrupt (PEP_x), whatever the state of the Endpoint x Interrupt Enable bit (PEP_x).

The user then has to acknowledge or to disable the interrupt source (e.g. USBHS_DEVEPTISRx.RXOUTI) or to clear the EPDISHDMA bit (by writing a one to the USBHS_DEVEPTIDRx.EPDISHDMAC bit) in order to complete the DMA transfer.

In Ping-pong mode, if the interrupt is associated to a new system-bank packet (e.g. Bank1) and the current DMA transfer is running on the previous packet (Bank0), then the previous-packet DMA transfer completes normally, but the new-packet DMA transfer does not start (not requested).

If the interrupt is not associated to a new system-bank packet (USBHS_DEVEPTISRx.NAKINI, NAKOUTI, etc.), then the request cancellation may occur at any time and may immediately pause the current DMA transfer.

This may be used for example to identify erroneous packets, to prevent them from being transferred into a buffer, to complete a DMA transfer by software after reception of a short packet, etc.

- **NYETDIS: NYET Token Disable**

0: Cleared when USBHS_DEVEPTIDRx.NYETDISC = 1. This enables the USBHS to handle the high-speed handshake following the USB 2.0 standard.

1: Set when USBHS_DEVEPTIERx.NYETDISS = 1. This sends a ACK handshake instead of a NYET handshake in High-speed mode.

- **RSTDT: Reset Data Toggle**

This bit is set when USBHS_DEVEPTIERx.RSTDTS = 1. This clears the data toggle sequence, i.e., sets to Data0 the data toggle sequence of the next sent (IN endpoints) or received (OUT endpoints) packet.

This bit is cleared instantaneously.

The user does not have to wait for this bit to be cleared.

- **STALLRQ: STALL Request**

0: Cleared when a new SETUP packet is received or when USBHS_DEVEPTIDRx.STALLRQC = 0.

1: Set when USBHS_DEVEPTIERx.STALLRQS = 1. This requests to send a STALL handshake to the host.

Downloaded from Arrow.com.

37.6.22 Device Endpoint x Mask Register (Isochronous Endpoints)

Name: USBHS_DEVEPTIMRx [x=0..9] (ISOENPT)

Address: 0x400381C0

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	RSTDT	–	EPDISHDMA
15	14	13	12	11	10	9	8
–	FIFOCON	KILLBK	NBUSYBKE	–	ERRORTRANSE	DATAXE	MDATAE
7	6	5	4	3	2	1	0
SHORTPACKETE	CRCERRE	OVERFE	HBISOFLUSHE	HBISOINERRE	UNDERFE	RXOUTE	TXINE

This register view is relevant only if EPTYPE = 0x1 in “Device Endpoint x Configuration Register” on page 616.

- **TXINE: Transmitted IN Data Interrupt**

0: Cleared when USBHS_DEVEPTIDRx.TXINEC = 1. This disables the Transmitted IN Data interrupt (USBHS_DEVEPTISRx.TXINI).

1: Set when USBHS_DEVEPTIERx.TXINES = 1. This enables the Transmitted IN Data interrupt (USBHS_DEVEPTISRx.TXINI).

- **RXOUTE: Received OUT Data Interrupt**

0: Cleared when USBHS_DEVEPTIDRx.RXOUTEC = 1. This disables the Received OUT Data interrupt (USBHS_DEVEPTISRx.RXOUTI).

1: Set when USBHS_DEVEPTIERx.RXOUTES = 1. This enables the Received OUT Data interrupt (USBHS_DEVEPTISRx.RXOUTI).

- **UNDERFE: Underflow Interrupt**

0: Cleared when USBHS_DEVEPTIDRx.UNDERFEC = 1. This disables the Underflow interrupt (USBHS_DEVEPTISRx.UNDERFI).

1: Set when USBHS_DEVEPTIERx.UNDERFES = 1. This enables the Underflow interrupt (USBHS_DEVEPTISRx.UNDERFI).

- **HBISOINERRE: High Bandwidth Isochronous IN Error Interrupt**

0: Cleared when the USBHS_DEVEPTIDRx.HBISOINERREC bit disables the HBISOINERRI interrupt.

1: Set when USBHS_DEVEPTIERx.HBISOINERRES = 1. This enables the HBISOINERRI interrupt.

- **HBISOFLUSHE: High Bandwidth Isochronous IN Flush Interrupt**

0: Cleared when the USBHS_DEVEPTIDRx.HBISOFLUSHEC bit disables the HBISOFLUSHI interrupt.

1: Set when USBHS_DEVEPTIERx.HBISOFLUSHES = 1. This enables the HBISOFLUSHI interrupt.

- **OVERFE: Overflow Interrupt**

0: Cleared when USBHS_DEVEPTIDRx.OVERFEC = 1. This disables the Overflow interrupt (USBHS_DEVEPTISRx.OVERFI).

1: Set when USBHS_DEVEPTIERx.OVERFES = 1. This enables the Overflow interrupt (USBHS_DEVEPTISRx.OVERFI).

Downloaded from Arrow.com.

- **CRCERRE: CRC Error Interrupt**

0: Cleared when USBHS_DEVEPTIDRx.CRCERREC = 1. This disables the CRC Error interrupt (USBHS_DEVEPTISRx.CRCERRI).

1: Set when USBHS_DEVEPTIERx.CRCERRES = 1. This enables the CRC Error interrupt (USBHS_DEVEPTISRx.CRCERRI).

- **SHORTPACKETE: Short Packet Interrupt**

0: Cleared when USBHS_DEVEPTIDRx.SHORTPACKETEC = 1. This disables the Short Packet interrupt (USBHS_DEVEPTISRx.SHORTPACKET).

1: Set when USBHS_DEVEPTIERx.SHORTPACKETES = 1. This enables the Short Packet interrupt (USBHS_DEVEPTISRx.SHORTPACKET).

If this bit is set for non-control IN endpoints, a short packet transmission is guaranteed upon ending a DMA transfer, thus signaling an end of isochronous frame or a bulk or interrupt end of transfer, provided that the End of DMA Buffer Output Enable (END_B_EN) bit and the Automatic Switch (AUTOSW) bit = 1.

- **MDATAE: MData Interrupt**

0: Cleared when USBHS_DEVEPTIDRx.MDATAEC = 1. This disables the Multiple DATA interrupt.

1: Set when the USBHS_DEVEPTIERx.MDATAES = 1. This enables the Multiple DATA interrupt (see DTSEQ bits).

- **DATAXE: DataX Interrupt**

0: Cleared when USBHS_DEVEPTIDRx.DATAXEC = 1. This disables the DATAX interrupt.

1: Set when the USBHS_DEVEPTIERx.DATAXES = 1. This enables the DATAX interrupt (see DTSEQ bits).

- **ERRORTRANSE: Transaction Error Interrupt**

0: Cleared when USBHS_DEVEPTIDRx.ERRORTRANSEC = 1. This disables the transaction error interrupt (USBHS_DEVEPTISRx.ERRORTRANS).

1: Set when USBHS_DEVEPTIERx.ERRORTRANSES = 1. This enables the transaction error interrupt (USBHS_DEVEPTISRx.ERRORTRANS).

- **NBUSYBKE: Number of Busy Banks Interrupt**

0: Cleared when USBHS_DEVEPTIDRx.NBUSYBKEC = 0. This disables the Number of Busy Banks interrupt (USBHS_DEVEPTISRx.NBUSYBK).

1: Set when USBHS_DEVEPTIERx.NBUSYBKES = 1. This enables the Number of Busy Banks interrupt (USBHS_DEVEPTISRx.NBUSYBK).

Downloaded from Arrow.com.

- **KILLBK: Kill IN Bank**

0: Cleared when the bank is killed.

1: Set when USBHS_DEVEPTIERx.KILLBKS = 1. This kills the last written bank.

Caution: The bank is really cleared when the “kill packet” procedure is accepted by the USBHS core. This bit is automatically cleared after the end of the procedure:

The bank is really killed: USBHS_DEVEPTISRx.NBUSYBK is decremented.

The bank is not cleared but sent (IN transfer): USBHS_DEVEPTISRx.NBUSYBK is decremented.

The bank is not cleared because it was empty.

The user should wait for this bit to be cleared before trying to kill another packet.

This kill request is refused if at the same time an IN token is coming and the last bank is the current one being sent on the USB line. If at least two banks are ready to be sent, there is no problem to kill a packet even if an IN token is coming. Indeed, in this case, the current bank is sent (IN transfer) while the last bank is killed.

- **FIFOCON: FIFO Control**

For control endpoints:

The FIFOCON and RWALL bits are irrelevant. Therefore, the software never uses them on these endpoints. When read, their value is always 0.

For IN endpoints:

0: Cleared (by writing a one to the USBHS_DEVEPTIDRx.FIFOCONC bit) to send the FIFO data and to switch to the next bank.

1: Set when the current bank is free, at the same time as USBHS_DEVEPTISRx.TXINI.

For OUT endpoints:

0: Cleared (by writing a one to the USBHS_DEVEPTIDRx.FIFOCONC bit) to free the current bank and to switch to the next bank.

1: Set when the current bank is full, at the same time as USBHS_DEVEPTISRx.RXOUTI.

- **EPDISHDMA: Endpoint Interrupts Disable HDMA Request**

This bit is set when USBHS_DEVEPTIERx.EPDISHDMAS = 1. This pauses the on-going DMA channel x transfer on any Endpoint x interrupt (PEP_x), whatever the state of the Endpoint x Interrupt Enable bit (PEP_x).

The user then has to acknowledge or to disable the interrupt source (e.g. USBHS_DEVEPTISRx.RXOUTI) or to clear the EPDISHDMA bit (by writing a one to the USBHS_DEVEPTIDRx.EPDISHDMAC bit) in order to complete the DMA transfer.

In Ping-pong mode, if the interrupt is associated to a new system-bank packet (e.g. Bank1) and the current DMA transfer is running on the previous packet (Bank0), then the previous-packet DMA transfer completes normally, but the new-packet DMA transfer does not start (not requested).

If the interrupt is not associated to a new system-bank packet (USBHS_DEVEPTISRx.NAKINI, NAKOUTI, etc.), then the request cancellation may occur at any time and may immediately pause the current DMA transfer.

This may be used for example to identify erroneous packets, to prevent them from being transferred into a buffer, to complete a DMA transfer by software after reception of a short packet, etc.

- **RSTDT: Reset Data Toggle**

This bit is set when USBHS_DEVEPTIERx.RSTDTS = 1. This clears the data toggle sequence, i.e., sets to Data0 the data toggle sequence of the next sent (IN endpoints) or received (OUT endpoints) packet.

This bit is cleared instantaneously.

The user does not have to wait for this bit to be cleared.

Downloaded from Arrow.com.

37.6.23 Device Endpoint x Disable Register (Control, Bulk, Interrupt Endpoints)

Name: USBHS_DEVEPTIDRx [x=0..9]

Address: 0x40038220

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	STALLRQC	–	NYETDISC	EPDISHDMAC
15	14	13	12	11	10	9	8
–	FIFOCONC	–	NBUSYBKEC	–	–	–	–
7	6	5	4	3	2	1	0
SHORT PACKETEC	STALLEDEC	OVERFEC	NAKINEC	NAKOUTEC	RXSTPEC	RXOUTEC	TXINEC

This register view is relevant only if EPTYPE = 0x0, 0x2 or 0x3 in “Device Endpoint x Configuration Register” on page 616.

For additional information, see “Device Endpoint x Mask Register (Control, Bulk, Interrupt Endpoints)” on page 630.

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Clears the corresponding bit in USBHS_DEVEPTIMRx.

- **TXINEC: Transmitted IN Interrupt Clear**
- **RXOUTEC: Received OUT Data Interrupt Clear**
- **RXSTPEC: Received SETUP Interrupt Clear**
- **NAKOUTEC: NAKed OUT Interrupt Clear**
- **NAKINEC: NAKed IN Interrupt Clear**
- **OVERFEC: Overflow Interrupt Clear**
- **STALLEDEC: STALLed Interrupt Clear**
- **SHORTPACKETEC: Shortpacket Interrupt Clear**
- **NBUSYBKEC: Number of Busy Banks Interrupt Clear**
- **FIFOCONC: FIFO Control Clear**
- **EPDISHDMAC: Endpoint Interrupts Disable HDMA Request Clear**
- **NYETDISC: NYET Token Disable Clear**
- **STALLRQC: STALL Request Clear**

Downloaded from Arrow.com.

37.6.24 Device Endpoint x Disable Register (Isochronous Endpoints)

Name: USBHS_DEVEPTIDRx [x=0..9] (ISOENPT)

Address: 0x40038220

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	EPDISHDMAC
15	14	13	12	11	10	9	8
–	FIFOCONC	–	NBUSYBKEC	–	ERRORTRANSEC	DATAXEC	MDATEC
7	6	5	4	3	2	1	0
SHORT PACKETEC	CRCERREC	OVERFEC	HBISOFLUSHEC	HBISOINERREC	UNDERFEC	RXOUTEC	TXINEC

This register view is relevant only if EPTYPE = 0x1 in “Device Endpoint x Configuration Register” on page 616.

For additional information, see “Device Endpoint x Mask Register (Isochronous Endpoints)” on page 633.

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Clears the corresponding bit in USBHS_DEVEPTIMRx.

- **TXINEC: Transmitted IN Interrupt Clear**
- **RXOUTEC: Received OUT Data Interrupt Clear**
- **UNDERFEC: Underflow Interrupt Clear**
- **HBISOINERREC: High Bandwidth Isochronous IN Error Interrupt Clear**
- **HBISOFLUSHEC: High Bandwidth Isochronous IN Flush Interrupt Clear**
- **OVERFEC: Overflow Interrupt Clear**
- **CRCERREC: CRC Error Interrupt Clear**
- **SHORTPACKETEC: Shortpacket Interrupt Clear**
- **MDATEC: MData Interrupt Clear**
- **DATAXEC: DataX Interrupt Clear**
- **ERRORTRANSEC: Transaction Error Interrupt Clear**
- **NBUSYBKEC: Number of Busy Banks Interrupt Clear**
- **FIFOCONC: FIFO Control Clear**
- **EPDISHDMAC: Endpoint Interrupts Disable HDMA Request Clear**

Downloaded from Arrow.com.

37.6.25 Device Endpoint x Enable Register (Control, Bulk, Interrupt Endpoints)

Name: USBHS_DEVEPTIERx [x=0..9]

Address: 0x400381F0

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	STALLRQS	RSTDTS	NYETDISS	EPDISHDMAS
15	14	13	12	11	10	9	8
–	FIFOCONS	KILLBKS	NBUSYBKES	–	–	–	–
7	6	5	4	3	2	1	0
SHORT PACKETES	STALLEDES	OVERFES	NAKINES	NAKOUTES	RXSTPES	RXOUTES	TXINES

This register view is relevant only if EPTYPE = 0x0, 0x2or 0x3 in “Device Endpoint x Configuration Register” on page 616.

For additional information, see “Device Endpoint x Mask Register (Control, Bulk, Interrupt Endpoints)” on page 630.

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Sets the corresponding bit in USBHS_DEVEPTIMRx.

- **TXINES: Transmitted IN Data Interrupt Enable**
- **RXOUTES: Received OUT Data Interrupt Enable**
- **RXSTPES: Received SETUP Interrupt Enable**
- **NAKOUTES: NAKed OUT Interrupt Enable**
- **NAKINES: NAKed IN Interrupt Enable**
- **OVERFES: Overflow Interrupt Enable**
- **STALLEDES: STALLed Interrupt Enable**
- **SHORTPACKETES: Short Packet Interrupt Enable**
- **NBUSYBKES: Number of Busy Banks Interrupt Enable**
- **KILLBKS: Kill IN Bank**
- **FIFOCONS: FIFO Control**
- **EPDISHDMAS: Endpoint Interrupts Disable HDMA Request Enable**
- **NYETDISS: NYET Token Disable Enable**
- **RSTDTS: Reset Data Toggle Enable**
- **STALLRQS: STALL Request Enable**

Downloaded from Arrow.com.

37.6.26 Device Endpoint x Enable Register (Isochronous Endpoints)

Name: USBHS_DEVEPTIERx [x=0..9] (ISOENPT)

Address: 0x400381F0

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	STALLRQS	RSTDTS	–	EPDISHDMAS
15	14	13	12	11	10	9	8
–	FIFOCONS	KILLBKS	NBUSYBKES	–	ERRORTRANSES	DATAXES	MDATAES
7	6	5	4	3	2	1	0
SHORT PACKETES	CRCERRES	OVERFES	HBISOFLUSHES	HBISOINERRES	UNDERFES	RXOUTES	TXINES

This register view is relevant only if EPTYPE = 0x1 in “Device Endpoint x Configuration Register” on page 616.

For additional information, see “Device Endpoint x Mask Register (Isochronous Endpoints)” on page 633.

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Clears the corresponding bit in USBHS_DEVEPTIMRx.

- **TXINES: Transmitted IN Data Interrupt Enable**
- **RXOUTES: Received OUT Data Interrupt Enable**
- **UNDERFES: Underflow Interrupt Enable**
- **HBISOINERRES: High Bandwidth Isochronous IN Error Interrupt Enable**
- **HBISOFLUSHES: High Bandwidth Isochronous IN Flush Interrupt Enable**
- **OVERFES: Overflow Interrupt Enable**
- **CRCERRES: CRC Error Interrupt Enable**
- **SHORTPACKETES: Short Packet Interrupt Enable**
- **MDATAES: MData Interrupt Enable**
- **DATAXES: DataX Interrupt Enable**
- **ERRORTRANSES: Transaction Error Interrupt Enable**
- **NBUSYBKES: Number of Busy Banks Interrupt Enable**
- **KILLBKS: Kill IN Bank**
- **FIFOCONS: FIFO Control**
- **EPDISHDMAS: Endpoint Interrupts Disable HDMA Request Enable**

Downloaded from Arrow.com.

- **RSTDTS: Reset Data Toggle Enable**
- **STALLRQS: STALL Request Enable**

Downloaded from Arrow.com.

37.6.27 Device DMA Channel x Next Descriptor Address Register

Name: USBHS_DEVDMANXTDSCx [x=1..7]

Address: 0x40038310 [1], 0x40038320 [2], 0x40038330 [3], 0x40038340 [4], 0x40038350 [5], 0x40038360 [6], 0x40038370 [7]

Access: Read/Write

31	30	29	28	27	26	25	24
NXT_DSC_ADD							
23	22	21	20	19	18	17	16
NXT_DSC_ADD							
15	14	13	12	11	10	9	8
NXT_DSC_ADD							
7	6	5	4	3	2	1	0
NXT_DSC_ADD							

- **NXT_DSC_ADD: Next Descriptor Address**

This field points to the next channel descriptor to be processed. This channel descriptor must be aligned, so bits 0 to 3 of the address must be equal to zero.

Downloaded from Arrow.com.

37.6.28 Device DMA Channel x Address Register

Name: USBHS_DEVDMAADDRESSx [x=1..7]

Address: 0x40038314 [1], 0x40038324 [2], 0x40038334 [3], 0x40038344 [4], 0x40038354 [5], 0x40038364 [6], 0x40038374 [7]

Access: Read/Write

31	30	29	28	27	26	25	24
BUFF_ADD							
23	22	21	20	19	18	17	16
BUFF_ADD							
15	14	13	12	11	10	9	8
BUFF_ADD							
7	6	5	4	3	2	1	0
BUFF_ADD							

- **BUFF_ADD: Buffer Address**

This field determines the AHB bus starting address of a DMA channel transfer.

Channel start and end addresses may be aligned on any byte boundary.

The firmware can write this field only when the USBHS_DEVDMASTATUS.CHANN_ENB bit is clear.

This field is updated at the end of the address phase of the current access to the AHB bus. It is incremented by the access byte width. The access width is 4 bytes (or less) at packet start or end, if the start or end address is not aligned on a word boundary.

The packet start address is either the channel start address or the next channel address to be accessed in the channel buffer. The packet end address is either the channel end address or the latest channel address accessed in the channel buffer.

The channel start address is written by software or loaded from the descriptor. The channel end address is either determined by the end of buffer or the USB device, or by the USB end of transfer if the USBHS_DEVDMACONTROLx.END_TR_EN bit is set.

Downloaded from Arrow.com.

37.6.29 Device DMA Channel x Control Register

Name: USBHS_DEVDMACONTROLx [x=1..7]

Address: 0x40038318 [1], 0x40038328 [2], 0x40038338 [3], 0x40038348 [4], 0x40038358 [5], 0x40038368 [6], 0x40038378 [7]

Access: Read/Write

31	30	29	28	27	26	25	24
BUFF_LENGTH							
23	22	21	20	19	18	17	16
BUFF_LENGTH							
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
BURST_LCK	DESC_LD_IT	END_BUFFIT	END_TR_IT	END_B_EN	END_TR_EN	LDNXT_DSC	CHANN_ENB

- **CHANN_ENB: Channel Enable Command**

0: The DMA channel is disabled at end of transfer and no transfer occurs upon request. This bit is also cleared by hardware when the channel source bus is disabled at end of buffer.

If the LDNXT_DSC bit has been cleared by descriptor loading, the firmware must set the corresponding CHANN_ENB bit to start the described transfer, if needed.

If the LDNXT_DSC bit is cleared, the channel is frozen and the channel registers may then be read and/or written reliably as soon as both USBHS_DEVDMASTATUS.CHANN_ENB and CHANN_ACT flags read as 0.

If a channel request is currently serviced when this bit is cleared, the DMA FIFO buffer is drained until it is empty, then the USBHS_DEVDMASTATUS.CHANN_ENB bit is cleared.

If the LDNXT_DSC bit is set at or after this bit clearing, then the currently loaded descriptor is skipped (no data transfer occurs) and the next descriptor is immediately loaded.

1: The USBHS_DEVDMASTATUS.CHANN_ENB bit is set, thus enabling the DMA channel data transfer. Then, any pending request starts the transfer. This may be used to start or resume any requested transfer.

- **LDNXT_DSC: Load Next Channel Transfer Descriptor Enable Command**

0: No channel register is loaded after the end of the channel transfer.

1: The channel controller loads the next descriptor after the end of the current transfer, i.e., when the USBHS_DEVDMASTATUS.CHANN_ENB bit is reset.

If the CHANN_ENB bit is cleared, the next descriptor is immediately loaded upon transfer request.

DMA Channel Control Command Summary:

Value LDNXT_DSC	Value CHANN_ENB	Name	Description
0	0	STOP_NOW	Stop now
0	1	RUN_AND_STOP	Run and stop at end of buffer
1	0	LOAD_NEXT_DESC	Load next descriptor now
1	1	RUN_AND_LINK	Run and link at end of buffer

Downloaded from Arrow.com.

- **END_TR_EN: End of Transfer Enable Control (OUT transfers only)**

0: The USB end of transfer is ignored.

1: The USBHS device can put an end to the current buffer transfer.

When set, a BULK or INTERRUPT short packet or the last packet of an ISOCHRONOUS (micro) frame (DATAX) closes the current buffer and the USBHS_DEVDMASTATUSx.END_TR_ST flag is raised.

This is intended for a USBHS non-prenegotiated end of transfer (BULK or INTERRUPT) or ISOCHRONOUS microframe data buffer closure.

- **END_B_EN: End of Buffer Enable Control**

0: DMA Buffer End has no impact on USB packet transfer.

1: The endpoint can validate the packet (according to the values programmed in the USBHS_DEVEPTCFGx.AUTOSW and USBHS_DEVEPTIERx.SHORTPACKETES fields) at DMA Buffer End, i.e., when USBHS_DEVDMASTATUS.BUFF_COUNT reaches 0.

This is mainly for short packet IN validations initiated by the DMA reaching end of buffer, but can be used for OUT packet truncation (discarding of unwanted packet data) at the end of DMA buffer.

- **END_TR_IT: End of Transfer Interrupt Enable**

0: USBHS device-initiated buffer transfer completion does not trigger any interrupt at USBHS_DEVDMASTATUSx.END_TR_ST rising.

1: An interrupt is sent after the buffer transfer is complete, if the USBHS device has ended the buffer transfer.

Use when the receive size is unknown.

- **END_BUFFIT: End of Buffer Interrupt Enable**

0: USBHS_DEVDMA_STATUSx.END_BF_ST rising does not trigger any interrupt.

1: An interrupt is generated when USBHS_HSTDMASTATUSx.BUFF_COUNT reaches zero.

- **DESC_LD_IT: Descriptor Loaded Interrupt Enable**

0: USBHS_DEVDMASTATUSx.DESC_LDST rising does not trigger any interrupt.

1: An interrupt is generated when a descriptor has been loaded from the bus.

- **BURST_LCK: Burst Lock Enable**

0: The DMA never locks bus access.

1: USB packets AHB data bursts are locked for maximum optimization of the bus bandwidth usage and maximization of fly-by AHB burst duration.

- **BUFF_LENGTH: Buffer Byte Length (Write-only)**

This field determines the number of bytes to be transferred until end of buffer. The maximum channel transfer size (32 KBytes) is reached when this field is 0 (default value). If the transfer size is unknown, this field should be set to 0, but the transfer end may occur earlier under USB device control.

When this field is written, the USBHS_DEVDMASTATUSx.BUFF_COUNT field is updated with the write value.

Notes:
1. Bits [31:2] are only writable when issuing a channel Control Command other than “Stop Now”.
2. For reliability, it is highly recommended to wait for both the USBHS_DEVDMASTATUSx.CHAN_ACT and the USBHS_DEVDMASTATUSx.CHAN_ENB flags to be at 0, thus ensuring the channel has been stopped before issuing a command other than “Stop Now”.

Downloaded from Arrow.com.

37.6.30 Device DMA Channel x Status Register

Name: USBHS_DEVDMASTATUSx [x=1..7]

Address: 0x4003831C [1], 0x4003832C [2], 0x4003833C [3], 0x4003834C [4], 0x4003835C [5], 0x4003836C [6], 0x4003837C [7]

Access: Read/Write

31	30	29	28	27	26	25	24
BUFF_COUNT							
23	22	21	20	19	18	17	16
BUFF_COUNT							
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	DESC_LDST	END_BF_ST	END_TR_ST	–	–	CHANN_ACT	CHANN_ENB

- **CHANN_ENB: Channel Enable Status**

0: If cleared, the DMA channel no longer transfers data, and may load the next descriptor if the USBHS_DEVDMACONTROLx.LDNXT_DSC bit is set.

When any transfer is ended either due to an elapsed byte count or to completion of a USBHS device-initiated transfer, this bit is automatically reset.

1: If set, the DMA channel is currently enabled and transfers data upon request.

This bit is normally set or cleared by writing into the USBHS_DEVDMACONTROLx.CHANN_ENB bit field either by software or descriptor loading.

If a channel request is currently serviced when the USBHS_DEVDMACONTROLx.CHANN_ENB bit is cleared, the DMA FIFO buffer is drained until it is empty, then this status bit is cleared.

- **CHANN_ACT: Channel Active Status**

0: The DMA channel is no longer trying to source the packet data.

When a packet transfer is ended, this bit is automatically reset.

1: The DMA channel is currently trying to source packet data, i.e., selected as the highest-priority requesting channel.

When a packet transfer cannot be completed due to an END_BF_ST, this flag stays set during the next channel descriptor load (if any) and potentially until completion of a USBHS packet transfer, if allowed by the new descriptor.

- **END_TR_ST: End of Channel Transfer Status**

0: Cleared automatically when read by software.

1: Set by hardware when the last packet transfer is complete, if the USBHS device has ended the transfer.

Valid until the CHANN_ENB flag is cleared at the end of the next buffer transfer.

- **END_BF_ST: End of Channel Buffer Status**

0: Cleared automatically when read by software.

1: Set by hardware when the BUFF_COUNT count-down reaches zero.

Valid until the CHANN_ENB flag is cleared at the end of the next buffer transfer.

Downloaded from Arrow.com.

- **DESC_LDST: Descriptor Loaded Status**

0: Cleared automatically when read by software.

1: Set by hardware when a descriptor has been loaded from the system bus.

Valid until the CHANN_ENB flag is cleared at the end of the next buffer transfer.

- **BUFF_COUNT: Buffer Byte Count**

This field determines the current number of bytes still to be transferred for this buffer.

This field is decremented from the AHB source bus access byte width at the end of this bus address phase.

The access byte width is 4 by default, or less, at DMA start or end, if the start or end address is not aligned on a word boundary.

At the end of buffer, the DMA accesses the USBHS device only for the number of bytes needed to complete it.

Note: For OUT endpoints, if the receive buffer byte length (BUFF_LENGTH) has been defaulted to zero because the USB transfer length is unknown, the actual buffer byte length received is 0x10000-BUFF_COUNT.

Downloaded from Arrow.com.

37.6.31 Host General Control Register

Name: USBHS_HSTCTRL

Address: 0x40038400

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	SPDCONF		–	RESUME	RESET	SOFE
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **SOFE: Start of Frame Generation Enable**

0: Disables the SOF generation and leaves the USB bus in idle state.

1: Generates SOF on the USB bus in Full- or High-speed mode and sends “keep alive” signals in Low-speed mode.

This bit is set when a USB reset is requested or an upstream resume interrupt is detected (USBHS_HSTISR.TXRSMI).

- **RESET: Send USB Reset**

0: No effect.

1: Generates a USB Reset on the USB bus.

This bit is cleared when the USB Reset has been sent.

It may be useful to write a zero to this bit when a device disconnection is detected (USBHS_HSTISR.DDISCI = 1) whereas a USB Reset is being sent.

- **RESUME: Send USB Resume**

0: No effect.

1: Generates a USB Resume on the USB bus.

This bit is cleared when the USB Resume has been sent or when a USB reset is requested.

This bit should be written to one only when the start of frame generation is enabled (SOFE = 1).

- **SPDCONF: Mode Configuration**

This field contains the host speed capability:.

Value	Name	Description
0	NORMAL	The host starts in Full-speed mode and performs a high-speed reset to switch to High-speed mode if the downstream peripheral is high-speed capable.
1	LOW_POWER	For a better consumption, if high speed is not needed.

Downloaded from Arrow.com.

37.6.32 Host Global Interrupt Status Register

Name: USBHS_HSTISR

Address: 0x40038404

Access: Read-only

31	30	29	28	27	26	25	24
DMA_7	DMA_6	DMA_5	DMA_4	DMA_3	DMA_2	DMA_1	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	PEP_9	PEP_8
15	14	13	12	11	10	9	8
PEP_7	PEP_6	PEP_5	PEP_4	PEP_3	PEP_2	PEP_1	PEP_0
7	6	5	4	3	2	1	0
–	HWUPI	HSOFI	RXRSMI	RSMEDI	RSTI	DDISCI	DCONNI

- **DCONNI: Device Connection Interrupt**

0: Cleared when USBHS_HSTICR.DCONNIC = 1.

1: Set when a new device has been connected to the USB bus.

- **DDISCI: Device Disconnection Interrupt**

0: Cleared when USBHS_HSTICR.DDISCIC = 1.

1: Set when the device has been removed from the USB bus.

- **RSTI: USB Reset Sent Interrupt**

0: Cleared when USBHS_HSTICR.RSTIC = 1.

1: Set when a USB Reset has been sent to the device.

- **RSMEDI: Downstream Resume Sent Interrupt**

0: Cleared when USBHS_HSTICR.RSMEDIC = 1.

1: Set when a Downstream Resume has been sent to the device.

- **RXRSMI: Upstream Resume Received Interrupt**

0: Cleared when USBHS_HSTICR.RXRSMIC = 1.

1: Set when an Upstream Resume has been received from the device.

- **HSOFI: Host Start of Frame Interrupt**

0: Cleared when USBHS_HSTICR.HSOFIC = 1.

1: Set when a SOF is issued by the host controller. This triggers a USB interrupt when HSOFE = 1. When using the host controller in Low-speed mode, this bit is also set when a keep-alive is sent.

- **HWUPI: Host Wake-Up Interrupt**

This bit is set when the host controller is in Suspend mode (SOFE = 0) and an upstream resume from the peripheral is detected.

This bit is set when the host controller is in Suspend mode (SOFE = 0) and a peripheral disconnection is detected.

This interrupt is generated even if the clock is frozen by the USBHS_CTRL.FRZCLK bit.

Downloaded from Arrow.com.

- **PEP_x: Pipe x Interrupt**

0: Cleared when the interrupt source is served.

1: Set when an interrupt is triggered by pipe x (USBHS_HSTPIPISRx). This triggers a USB interrupt if the corresponding bit in USBHS_HSTIMR = 1.

- **DMA_x: DMA Channel x Interrupt**

0: Cleared when the USBHS_HSTDMASTATUSx interrupt source is cleared.

1: Set when an interrupt is triggered by the DMA channel x. This triggers a USB interrupt if the corresponding bit in USBHS_HSTIMR = 1.

Downloaded from Arrow.com.

37.6.33 Host Global Interrupt Clear Register

Name: USBHS_HSTICR

Address: 0x40038408

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	HWUPIC	HSOFIC	RXRSMIC	RSMEDIC	RSTIC	DDISCIC	DCONNIC

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Clears the corresponding bit in USBHS_HSTISR.

- **DCONNIC: Device Connection Interrupt Clear**
- **DDISCIC: Device Disconnection Interrupt Clear**
- **RSTIC: USB Reset Sent Interrupt Clear**
- **RSMEDIC: Downstream Resume Sent Interrupt Clear**
- **RXRSMIC: Upstream Resume Received Interrupt Clear**
- **HSOFIC: Host Start of Frame Interrupt Clear**
- **HWUPIC: Host Wake-Up Interrupt Clear**

Downloaded from Arrow.com.

37.6.34 Host Global Interrupt Set Register

Name: USBHS_HSTIFR

Address: 0x4003840C

Access: Write-only

31	30	29	28	27	26	25	24
DMA_7	DMA_6	DMA_5	DMA_4	DMA_3	DMA_2	DMA_1	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	HWUPIS	HSOFIS	RXRSMIS	RSMEDIS	RSTIS	DDISCIS	DCONNIS

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Sets the corresponding bit in USBHS_HSTISR, which may be useful for test or debug purposes.

- **DCONNIS: Device Connection Interrupt Set**
- **DDISCIS: Device Disconnection Interrupt Set**
- **RSTIS: USB Reset Sent Interrupt Set**
- **RSMEDIS: Downstream Resume Sent Interrupt Set**
- **RXRSMIS: Upstream Resume Received Interrupt Set**
- **HSOFIS: Host Start of Frame Interrupt Set**
- **HWUPIS: Host Wake-Up Interrupt Set**
- **DMA_x: DMA Channel x Interrupt Set**

Downloaded from Arrow.com.

37.6.35 Host Global Interrupt Mask Register

Name: USBHS_HSTIMR

Address: 0x40038410

Access: Read-only

31	30	29	28	27	26	25	24
DMA_7	DMA_6	DMA_5	DMA_4	DMA_3	DMA_2	DMA_1	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	PEP_9	PEP_8
15	14	13	12	11	10	9	8
PEP_7	PEP_6	PEP_5	PEP_4	PEP_3	PEP_2	PEP_1	PEP_0
7	6	5	4	3	2	1	0
–	HWUPIE	HSOFIE	RXRSMIE	RSMEDIE	RSTIE	DDISCIE	DCONNIE

- **DCONNIE: Device Connection Interrupt Enable**

0: Cleared when USBHS_HSTIDR.DCONNIEC = 1. This disables the Device Connection interrupt (USBHS_HSTISR.DCONNI).

1: Set when USBHS_HSTIER.DCONNIES = 1. This enables the Device Connection interrupt (USBHS_HSTISR.DCONNI).

- **DDISCIE: Device Disconnection Interrupt Enable**

0: Cleared when USBHS_HSTIDR.DDISCIEC = 1. This disables the Device Disconnection interrupt (USBHS_HSTISR.DDISCI).

1: Set when USBHS_HSTIER.DDISCIES = 1. This enables the Device Disconnection interrupt (USBHS_HSTISR.DDISCI).

- **RSTIE: USB Reset Sent Interrupt Enable**

0: Cleared when USBHS_HSTIDR.RSTIEC = 1. This disables the USB Reset Sent interrupt (USBHS_HSTISR.RSTI).

1: Set when USBHS_HSTIER.RSTIES = 1. This enables the USB Reset Sent interrupt (USBHS_HSTISR.RSTI).

- **RSMEDIE: Downstream Resume Sent Interrupt Enable**

0: Cleared when USBHS_HSTIDR.RSMEDIEC = 1. This disables the Downstream Resume interrupt (USBHS_HSTISR.RSMEDI).

1: Set when USBHS_HSTIER.RSMEDIES = 1. This enables the Downstream Resume interrupt (USBHS_HSTISR.RSMEDI).

- **RXRSMIE: Upstream Resume Received Interrupt Enable**

0: Cleared when USBHS_HSTIDR.RXRSMIEC= 1. This disables the Downstream Resume interrupt (USBHS_HSTISR.RXRSMI).

1: Set when USBHS_HSTIER.RXRSMIES = 1. This enables the Upstream Resume Received interrupt (USBHS_HSTISR.RXRSMI).

- **HSOFIE: Host Start of Frame Interrupt Enable**

0: Cleared when USBHS_HSTIDR.HSOFIEC = 1. This disables the Host Start of Frame interrupt (USBHS_HSTISR.HSOFI).

1: Set when USBHS_HSTIER.HSOFIES= 1. This enables the Host Start of Frame interrupt (USBHS_HSTISR.HSOFI).

Downloaded from Arrow.com.

- **HWUPIE: Host Wake-Up Interrupt Enable**

0: Cleared when USBHS_HSTIDR.HWUPIEC = 1. This disables the Host Wake-up Interrupt (USBHS_HSTISR.HWUPI).

1: Set when USBHS_HSTIER.HWUPIES = 1. This enables the Host Wake-up Interrupt (USBHS_HSTISR.HWUPI).

- **PEP_x: Pipe x Interrupt Enable**

0: Cleared when PEP_x = 1. This disables the Pipe x Interrupt (PEP_x).

1: Set when the corresponding bit in USBHS_HSTIER = 1. This enables the Pipe x Interrupt (USBHS_HSTISR.PEP_x).

- **DMA_x: DMA Channel x Interrupt Enable**

0: Cleared when the corresponding bit in USBHS_HSTIDR = 1. This disables the DMA Channel x Interrupt (USBHS_HSTISR.DMA_x).

1: Set when the corresponding bit in USBHS_HSTIER = 1. This enables the DMA Channel x Interrupt (USBHS_HSTISR.DMA_x).

Downloaded from Arrow.com.

37.6.36 Host Global Interrupt Disable Register

Name: USBHS_HSTIDR

Address: 0x40038414

Access: Write-only

31	30	29	28	27	26	25	24
DMA_7	DMA_6	DMA_5	DMA_4	DMA_3	DMA_2	DMA_1	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	PEP_9	PEP_8
15	14	13	12	11	10	9	8
PEP_7	PEP_6	PEP_5	PEP_4	PEP_3	PEP_2	PEP_1	PEP_0
7	6	5	4	3	2	1	0
–	HWUPIEC	HSOFIEC	RXRSMIEC	RSMEDIEC	RSTIEC	DDISCIEC	DCONNIEC

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Clears the corresponding bit in USBHS_HSTIMR.

- **DCONNIEC: Device Connection Interrupt Disable**
- **DDISCIEC: Device Disconnection Interrupt Disable**
- **RSTIEC: USB Reset Sent Interrupt Disable**
- **RSMEDIEC: Downstream Resume Sent Interrupt Disable**
- **RXRSMIEC: Upstream Resume Received Interrupt Disable**
- **HSOFIEC: Host Start of Frame Interrupt Disable**
- **HWUPIEC: Host Wake-Up Interrupt Disable**
- **PEP_x: Pipe x Interrupt Disable**
- **DMA_x: DMA Channel x Interrupt Disable**

Downloaded from Arrow.com.

37.6.37 Host Global Interrupt Enable Register

Name: USBHS_HSTIER

Address: 0x40038418

Access: Write-only

31	30	29	28	27	26	25	24
DMA_7	DMA_6	DMA_5	DMA_4	DMA_3	DMA_2	DMA_1	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	PEP_9	PEP_8
15	14	13	12	11	10	9	8
PEP_7	PEP_6	PEP_5	PEP_4	PEP_3	PEP_2	PEP_1	PEP_0
7	6	5	4	3	2	1	0
–	HWUPIES	HSOFIES	RXRSMIES	RSMEDIES	RSTIES	DDISCIES	DCONNIES

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Sets the corresponding bit in USBHS_HSTISR.

- **DCONNIES: Device Connection Interrupt Enable**
- **DDISCIES: Device Disconnection Interrupt Enable**
- **RSTIES: USB Reset Sent Interrupt Enable**
- **RSMEDIES: Downstream Resume Sent Interrupt Enable**
- **RXRSMIES: Upstream Resume Received Interrupt Enable**
- **HSOFIES: Host Start of Frame Interrupt Enable**
- **HWUPIES: Host Wake-Up Interrupt Enable**
- **PEP_x: Pipe x Interrupt Enable**
- **DMA_x: DMA Channel x Interrupt Enable**

Downloaded from Arrow.com.

37.6.38 Host Frame Number Register

Name: USBHS_HSTFNUM

Address: 0x40038420

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
FLENHIGH							
15	14	13	12	11	10	9	8
–	–	FNUM					
7	6	5	4	3	2	1	0
FNUM					MFNUM		

- **MFNUM: Micro Frame Number**

This field contains the current microframe number (can vary from 0 to 7), updated every 125 µs.

When operating in Full-speed mode, this field is tied to zero.

- **FNUM: Frame Number**

This field contains the current SOF number.

This field can be written. In this case, the MFNUM field is reset to zero.

- **FLENHIGH: Frame Length**

In High-speed mode, this field contains the 8 high-order bits of the 16-bit internal frame counter (at 30 MHz, the counter length is 3750 to ensure a SOF generation every 125 µs).

Downloaded from Arrow.com.

37.6.39 Host Address 1 Register

Name: USBHS_HSTADDR1

Address: 0x40038424

Access: Read/Write

31	30	29	28	27	26	25	24
–	HSTADDRP3						
23	22	21	20	19	18	17	16
–	HSTADDRP2						
15	14	13	12	11	10	9	8
–	HSTADDRP1						
7	6	5	4	3	2	1	0
–	HSTADDRP0						

- **HSTADDRP0: USB Host Address**

This field contains the address of the Pipe0 of the USB device.

This field is cleared when a USB reset is requested.

- **HSTADDRP1: USB Host Address**

This field contains the address of the Pipe1 of the USB device.

This field is cleared when a USB reset is requested.

- **HSTADDRP2: USB Host Address**

This field contains the address of the Pipe2 of the USB device.

This field is cleared when a USB reset is requested.

- **HSTADDRP3: USB Host Address**

This field contains the address of the Pipe3 of the USB device.

This field is cleared when a USB reset is requested.

Downloaded from Arrow.com.

37.6.40 Host Address 2 Register

Name: USBHS_HSTADDR2

Address: 0x40038428

Access: Read/Write

31	30	29	28	27	26	25	24
–	HSTADDRP7						
23	22	21	20	19	18	17	16
–	HSTADDRP6						
15	14	13	12	11	10	9	8
–	HSTADDRP5						
7	6	5	4	3	2	1	0
–	HSTADDRP4						

- **HSTADDRP4: USB Host Address**

This field contains the address of the Pipe4 of the USB device.

This field is cleared when a USB reset is requested.

- **HSTADDRP5: USB Host Address**

This field contains the address of the Pipe5 of the USB device.

This field is cleared when a USB reset is requested.

- **HSTADDRP6: USB Host Address**

This field contains the address of the Pipe6 of the USB device.

This field is cleared when a USB reset is requested.

- **HSTADDRP7: USB Host Address**

This field contains the address of the Pipe7 of the USB device.

This field is cleared when a USB reset is requested.

Downloaded from Arrow.com.

37.6.41 Host Address 3 Register

Name: USBHS_HSTADDR3

Address: 0x4003842C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	HSTADDRP9						
7	6	5	4	3	2	1	0
–	HSTADDRP8						

- **HSTADDRP8: USB Host Address**

This field contains the address of the Pipe8 of the USB device.

This field is cleared when a USB reset is requested.

- **HSTADDRP9: USB Host Address**

This field contains the address of the Pipe9 of the USB device.

This field is cleared when a USB reset is requested.

Downloaded from Arrow.com.

37.6.42 Host Pipe Register

Name: USBHS_HSTPIP

Address: 0x4003841C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	PRST8
23	22	21	20	19	18	17	16
PRST7	PRST6	PRST5	PRST4	PRST3	PRST2	PRST1	PRST0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	PEN8
7	6	5	4	3	2	1	0
PEN7	PEN6	PEN5	PEN4	PEN3	PEN2	PEN1	PEN0

- **PENx: Pipe x Enable**

0: Disables Pipe x, which forces the Pipe x state to inactive and resets the pipe x registers (USBHS_HSTPIPCFGx, USBHS_HSTPIPISRx, USBHS_HSTPIPIMRx), but not the pipe configuration (USBHS_HSTPIPCFGx.ALLOC, USBHS_HSTPIPCFGx.PBK, USBHS_HSTPIPCFGx.PSIZE).

1: Enables Pipe x.

- **PRSTx: Pipe x Reset**

0: Completes the reset operation and allows to start using the FIFO.

1: Resets the Pipe x FIFO. This resets the pipe x registers (USBHS_HSTPIPCFGx, USBHS_HSTPIPISRx, USBHS_HSTPIPIMRx), but not the pipe configuration (ALLOC, PBK, PSIZE, PTOKEN, PTYPE, PEPNUM, INTFRQ). The whole pipe mechanism (FIFO counter, reception, transmission, etc.) is reset, apart from the Data Toggle management. The pipe configuration remains active and the pipe is still enabled.

Downloaded from Arrow.com.

37.6.43 Host Pipe x Configuration Register

Name: USBHS_HSTPIPCFGx [x=0..9]

Address: 0x40038500

Access: Read/Write

31	30	29	28	27	26	25	24
INTFRQ							
23	22	21	20	19	18	17	16
–	–	–	–	PEPNUM			
15	14	13	12	11	10	9	8
–	–	PTYPE		–	AUTOSW	PTOKEN	
7	6	5	4	3	2	1	0
–	PSIZE			PBK		ALLOC	–

For High-speed Bulk-out Pipe, see “Host Pipe x Configuration Register (High-speed Bulk-out or High-speed Control Pipe)” on page 664.

- **ALLOC: Pipe Memory Allocate**

0: Frees the pipe memory.

1: Allocates the pipe memory.

This bit is cleared when a USB Reset is requested.

Refer to Section 37.5.1.5 ”DPRAM Management” for more details.

- **PBK: Pipe Banks**

This field contains the number of banks for the pipe.

Value	Name	Description
0	1_BANK	Single-bank pipe
1	2_BANK	Double-bank pipe
2	3_BANK	Triple-bank pipe
3	–	Reserved

For control pipes, a single-bank pipe (0b00) should be selected.

This field is cleared upon sending a USB reset.

Downloaded from Arrow.com.

- **PSIZE: Pipe Size**

This field contains the size of each pipe bank.

Value	Name	Description
0	8_BYTE	8 bytes
1	16_BYTE	16 bytes
2	32_BYTE	32 bytes
3	64_BYTE	64 bytes
4	128_BYTE	128 bytes
5	256_BYTE	256 bytes
6	512_BYTE	512 bytes
7	1024_BYTE	1024 bytes

This field is cleared upon sending a USB reset.

- **PTOKEN: Pipe Token**

This field contains the pipe token.

Value	Name	Description
0	SETUP	SETUP
1	IN	IN
2	OUT	OUT
3	–	Reserved

- **AUTOSW: Automatic Switch**

This bit is cleared upon sending a USB reset.

0: The automatic bank switching is disabled.

1: The automatic bank switching is enabled.

- **PTYPE: Pipe Type**

This field contains the pipe type.

Value	Name	Description
0	CTRL	Control
1	ISO	Isochronous
2	BLK	Bulk
3	INTRPT	Interrupt

This field is cleared upon sending a USB reset.

- **PEPNUM: Pipe Endpoint Number**

This field contains the number of the endpoint targeted by the pipe. This value is from 0 to 10.

This field is cleared upon sending a USB reset.

Downloaded from Arrow.com.

- **INTFRQ: Pipe Interrupt Request Frequency**

This field contains the maximum value in milliseconds of the polling period for an Interrupt Pipe.

This value has no effect for a non-Interrupt Pipe.

This field is cleared upon sending a USB reset.

Downloaded from Arrow.com.

37.6.44 Host Pipe x Configuration Register (High-speed Bulk-out or High-speed Control Pipe)

Name: USBHS_HSTPIPCFGx [x=0..9] (HSBOHSCP)

Address: 0x40038500

Access: Read/Write

31	30	29	28	27	26	25	24
BINTERVAL							
23	22	21	20	19	18	17	16
–	–	–	PINGEN	PEPNUM			
15	14	13	12	11	10	9	8
–	–	PTYPE		–	AUTOSW	PTOKEN	
7	6	5	4	3	2	1	0
–	PSIZE			PBK		ALLOC	–

This configuration is relevant only if PTYPE = 0x0 or 0x2 in “Host Pipe x Configuration Register” on page 661.

- **ALLOC: Pipe Memory Allocate**

0: Frees the pipe memory.

1: Allocates the pipe memory.

This bit is cleared when a USB Reset is requested.

Refer to Section 37.5.1.5 ”DPRAM Management” for more details.

- **PBK: Pipe Banks**

This field contains the number of banks for the pipe.

Value	Name	Description
0	1_BANK	Single-bank pipe
1	2_BANK	Double-bank pipe
2	3_BANK	Triple-bank pipe
3	–	Reserved

For control pipes, a single-bank pipe (0b00) should be selected.

This field is cleared upon sending a USB reset.

Downloaded from Arrow.com.

- **PSIZE: Pipe Size**

This field contains the size of each pipe bank.

Value	Name	Description
0	8_BYTE	8 bytes
1	16_BYTE	16 bytes
2	32_BYTE	32 bytes
3	64_BYTE	64 bytes
4	128_BYTE	128 bytes
5	256_BYTE	256 bytes
6	512_BYTE	512 bytes
7	1024_BYTE	1024 bytes

This field is cleared upon sending a USB reset.

- **PTOKEN: Pipe Token**

This field contains the pipe token.

Value	Name	Description
0	SETUP	SETUP
1	IN	IN
2	OUT	OUT
3	–	Reserved

- **AUTOSW: Automatic Switch**

This bit is cleared upon sending a USB reset.

0: The automatic bank switching is disabled.

1: The automatic bank switching is enabled.

- **PTYPE: Pipe Type**

This field contains the pipe type.

Value	Name	Description
0	CTRL	Control
1	–	Reserved
2	BLK	Bulk
3	–	Reserved

This field is cleared upon sending a USB reset.

- **PEPNUM: Pipe Endpoint Number**

This field contains the number of the endpoint targeted by the pipe. This value is from 0 to 10.

This field is cleared upon sending a USB reset.

Downloaded from Arrow.com.

- **PINGEN: Ping Enable**

This bit is relevant for High-speed Bulk-out transaction only (including the control data stage and the control status stage).

0: Disables the ping protocol.

1: Enables the ping mechanism according to the USB 2.0 Standard.

This bit is cleared upon sending a USB reset.

- **BINTERVAL: Binterval Parameter for the Bulk-Out/Ping Transaction**

This field contains the Ping/Bulk-out period.

- If BINTERVAL > 0 and PINGEN = 1, one PING token is sent every BINTERVAL microframe until it is ACKed by the peripheral.
- If BINTERVAL = 0 and PINGEN = 1, multiple consecutive PING tokens are sent in the same microframe until they are ACKed.
- If BINTERVAL > 0 and PINGEN = 0, one OUT token is sent every BINTERVAL microframe until it is ACKed by the peripheral.
- If BINTERVAL = 0 and PINGEN = 0, multiple consecutive OUT tokens are sent in the same microframe until they are ACKed.

This value must be in the range from 0 to 255.

Downloaded from Arrow.com.

37.6.45 Host Pipe x Status Register (Control, Bulk Pipes)

Name: USBHS_HSTPIPISRx [x=0..9]

Address: 0x40038530

Access: Read-only

31	30	29	28	27	26	25	24
–	PBYCT						
23	22	21	20	19	18	17	16
PBYCT				–	CFGOK	–	RWALL
15	14	13	12	11	10	9	8
CURRBK		NBUSYBK		–	–	DTSEQ	
7	6	5	4	3	2	1	0
SHORTPACKETI	RXSTALLDI	OVERFI	NAKEDI	PERRI	TXSTPI	TXOUTI	RXINI

This register view is relevant only if PTYPE = 0x0 or 0x2 in “Host Pipe x Configuration Register” on page 661.

- **RXINI: Received IN Data Interrupt**

0: Cleared when USBHS_HSTPIPICR.RXINIC = 1.

1: Set when a new USB message is stored in the current bank of the pipe. This triggers an interrupt if USBHS_HSTPIPIMR.RXINE = 1.

- **TXOUTI: Transmitted OUT Data Interrupt**

0: Cleared when USBHS_HSTPIPICR.TXOUTIC = 1.

1: Set when the current OUT bank is free and can be filled. This triggers an interrupt if USBHS_HSTPIPIMR.TXOUTE = 1.

- **TXSTPI: Transmitted SETUP Interrupt**

0: Cleared when USBHS_HSTPIPICR.TXSTPIC = 1.

1: Set, for control pipes, when the current SETUP bank is free and can be filled. This triggers an interrupt if USBHS_HSTPIPIMR.TXSTPE = 1.

- **PERRI: Pipe Error Interrupt**

0: Cleared when the error source bit is cleared.

1: Set when an error occurs on the current bank of the pipe. This triggers an interrupt if the USBHS_HSTPIPIMR.PERRE bit is set. Refer to the USBHS_HSTPIPERRx register to determine the source of the error.

- **NAKEDI: NAKed Interrupt**

0: Cleared when USBHS_HSTPIPICR.NAKEDIC = 1.

1: Set when a NAK has been received on the current bank of the pipe. This triggers an interrupt if USBHS_HSTPIPIMR.NAKEDE = 1.

- **OVERFI: Overflow Interrupt**

0: Cleared when USBHS_HSTPIPICR.OVERFIC = 1.

1: Set when the current pipe has received more data than the maximum length of the current pipe. An interrupt is triggered if USBHS_HSTPIPIMR.OVERFIE = 1.

Downloaded from Arrow.com.

- **RXSTALLDI: Received STALLed Interrupt**

This bit is set when a STALL handshake has been received on the current bank of the pipe. The pipe is automatically frozen. This triggers an interrupt if USBHS_HSTPIPIMR.RXSTALLE = 1.

0: Cleared when USBHS_HSTPIPICR.RXSTALLDIC = 1.

- **SHORTPACKETI: Short Packet Interrupt**

0: Cleared when USBHS_HSTPIPICR.SHORTPACKETIC = 1.

1: Set when a short packet is received by the host controller (packet length inferior to the PSIZE programmed field).

- **DTSEQ: Data Toggle Sequence**

This field indicates the data PID of the current bank.

Value	Name	Description
0	DATA0	Data0 toggle sequence
1	DATA1	Data1 toggle sequence
2	–	Reserved
3	–	Reserved

For an OUT pipe, this field indicates the data toggle of the next packet that is to be sent.

For an IN pipe, this field indicates the data toggle of the received packet stored in the current bank.

- **NBUSYBK: Number of Busy Banks**

This field indicates the number of busy banks.

For an OUT pipe, this field indicates the number of busy banks, filled by the user, ready for OUT transfer. When all banks are busy, this triggers a PEP_x interrupt if USBHS_HSTPIPIMRx.NBUSYBKE = 1.

For an IN pipe, this field indicates the number of busy banks filled by IN transaction from the Device. When all banks are free, this triggers a PEP_x interrupt if USBHS_HSTPIPIMRx.NBUSYBKE = 1.

Value	Name	Description
0	0_BUSY	0 busy bank (all banks free)
1	1_BUSY	1 busy bank
2	2_BUSY	2 busy banks
3	3_BUSY	3 busy banks

- **CURRBK: Current Bank**

For non-control pipe, this field indicates the number of the current bank.

Value	Name	Description
0	BANK0	Current bank is bank0
1	BANK1	Current bank is bank1
2	BANK2	Current bank is bank2
3	–	Reserved

This field may be updated 1 clock cycle after the RWALL bit changes, so the user should not poll it as an interrupt bit.

Downloaded from Arrow.com.

- **RWALL: Read/Write Allowed**

For an OUT pipe, this bit is set when the current bank is not full, i.e., the software can write further data into the FIFO.

For an IN pipe, this bit is set when the current bank is not empty, i.e., the software can read further data from the FIFO.

This bit is cleared otherwise.

This bit is also cleared when the RXSTALLDI or the PERRI bit = 1.

- **CFGOK: Configuration OK Status**

This bit is set/cleared when the USBHS_HSTPIPCFGx.ALLOC bit is set.

This bit is set if the pipe x number of banks (USBHS_HSTPIPCFGx.PBK) and size (USBHS_HSTPIPCFGx.PSIZE) are correct compared to the maximal allowed number of banks and size for this pipe and to the maximal FIFO size (i.e., the DPRAM size).

If this bit is cleared, the user should rewrite correct values for the PBK and PSIZE fields in the USBHS_HSTPIPCFGx register.

- **PBYCT: Pipe Byte Count**

This field contains the byte count of the FIFO.

For an OUT pipe, the field is incremented after each byte written by the user into the pipe and decremented after each byte sent to the peripheral.

For an IN pipe, the field is incremented after each byte received from the peripheral and decremented after each byte read by the user from the pipe.

This field may be updated 1 clock cycle after the RWALL bit changes, so the user should not poll this field as an interrupt bit.

Downloaded from Arrow.com.

37.6.46 Host Pipe x Status Register (Interrupt Pipes)

Name: USBHS_HSTPIPISRx [x=0..9] (INTPIPES)

Address: 0x40038530

Access: Read-only

31	30	29	28	27	26	25	24
–	PBYCT						
23	22	21	20	19	18	17	16
PBYCT				–	CFGOK	–	RWALL
15	14	13	12	11	10	9	8
CURRBK		NBUSYBK		–	–	DTSEQ	
7	6	5	4	3	2	1	0
SHORTPACKETI	RXSTALLDI	OVERFI	NAKEDI	PERRI	UNDERFI	TXOUTI	RXINI

This register view is relevant only if PTYPE = 0x3 in “Host Pipe x Configuration Register” on page 661.

- **RXINI: Received IN Data Interrupt**

0: Cleared when USBHS_HSTPIPICR.RXINIC = 1.

1: Set when a new USB message is stored in the current bank of the pipe. This triggers an interrupt if the USBHS_HSTPIPIMR.RXINE bit = 1.

- **TXOUTI: Transmitted OUT Data Interrupt**

0: Cleared when USBHS_HSTPIPICR.TXOUTIC = 1.

1: Set when the current OUT bank is free and can be filled. This triggers an interrupt if USBHS_HSTPIPIMR.TXOUTE = 1.

- **UNDERFI: Underflow Interrupt**

This bit is set, for an isochronous and interrupt IN/OUT pipe, when an error flow occurs. This triggers an interrupt if UNDERFIE = 1.

This bit is set, for an isochronous or interrupt OUT pipe, when a transaction underflow occurs in the current pipe (the pipe cannot send the OUT data packet in time because the current bank is not ready). A zero-length-packet (ZLP) is sent instead.

This bit is set, for an isochronous or interrupt IN pipe, when a transaction flow error occurs in the current pipe, i.e, the current bank of the pipe is not free while a new IN USB packet is received. This packet is not stored in the bank. For an interrupt pipe, the overflowed packet is ACKed to comply with the USB standard.

This bit is cleared when USBHS_HSTPIPICR.UNDERFIEC = 1.

- **PERRI: Pipe Error Interrupt**

0: Cleared when the error source bit is cleared.

1: Set when an error occurs on the current bank of the pipe. This triggers an interrupt if the USBHS_HSTPIPIMR.PERRE bit is set. Refer to the USBHS_HSTPIPERRx register to determine the source of the error.

- **NAKEDI: NAKed Interrupt**

0: Cleared when USBHS_HSTPIPICR.NAKEDIC = 1.

1: Set when a NAK has been received on the current bank of the pipe. This triggers an interrupt if the USBHS_HSTPIPIMR.NAKEDE bit = 1.

Downloaded from Arrow.com.

- **OVERFI: Overflow Interrupt**

0: Cleared when USBHS_HSTPIPICR.OVERFIC = 1.

1: Set when the current pipe has received more data than the maximum length of the current pipe. An interrupt is triggered if the USBHS_HSTPIPIMR.OVERFIE bit = 1.

- **RXSTALLDI: Received STALLed Interrupt**

0: Cleared when USBHS_HSTPIPICR.RXSTALLDIC = 1.

1: Set when a STALL handshake has been received on the current bank of the pipe. The pipe is automatically frozen. This triggers an interrupt if USBHS_HSTPIPIMR.RXSTALLE = 1.

- **SHORTPACKETI: Short Packet Interrupt**

0: Cleared when USBHS_HSTPIPICR.SHORTPACKETIC = 1.

1: Set when a short packet is received by the host controller (packet length inferior to the PSIZE programmed field).

- **DTSEQ: Data Toggle Sequence**

This field indicates the data PID of the current bank.

Value	Name	Description
0	DATA0	Data0 toggle sequence
1	DATA1	Data1 toggle sequence
2	–	Reserved
3	–	Reserved

For an OUT pipe, this field indicates the data toggle of the next packet that is to be sent.

For an IN pipe, this field indicates the data toggle of the received packet stored in the current bank.

- **NBUSYBK: Number of Busy Banks**

This field indicates the number of busy banks.

For an OUT pipe, this field indicates the number of busy banks, filled by the user, ready for an OUT transfer. When all banks are busy, this triggers a PEP_x interrupt if USBHS_HSTPIPIMRx.NBUSYBKE = 1.

For an IN pipe, this field indicates the number of busy banks filled by IN transaction from the device. When all banks are free, this triggers a PEP_x interrupt if USBHS_HSTPIPIMRx.NBUSYBKE = 1.

Value	Name	Description
0	0_BUSY	0 busy bank (all banks free)
1	1_BUSY	1 busy bank
2	2_BUSY	2 busy banks
3	3_BUSY	3 busy banks

Downloaded from Arrow.com.

- **CURRBK: Current Bank**

For a non-control pipe, this field indicates the number of the current bank.

Value	Name	Description
0	BANK0	Current bank is bank0
1	BANK1	Current bank is bank1
2	BANK2	Current bank is bank2
3	–	Reserved

This field may be updated 1 clock cycle after the RWALL bit changes, so the user should not poll it as an interrupt bit.

- **RWALL: Read/Write Allowed**

For an OUT pipe, this bit is set when the current bank is not full, i.e., the software can write further data into the FIFO.

For an IN pipe, this bit is set when the current bank is not empty, i.e., the software can read further data from the FIFO.

This bit is cleared otherwise.

This bit is also cleared when RXSTALLDI or PERRI = 1.

- **CFGOK: Configuration OK Status**

This bit is set/cleared when the USBHS_HSTPIPCFGx.ALLOC bit is set.

This bit is set if the pipe x number of banks (USBHS_HSTPIPCFGx.PBK) and size (USBHS_HSTPIPCFGx.PSIZE) are correct compared to the maximal allowed number of banks and size for this pipe, and to the maximal FIFO size (i.e., the DPRAM size).

If this bit is cleared, the user should rewrite correct values for the PBK and PSIZE fields in the USBHS_HSTPIPCFGx register.

- **PBYCT: Pipe Byte Count**

This field contains the byte count of the FIFO.

For an OUT pipe, the field is incremented after each byte written by the user into the pipe and decremented after each byte sent to the peripheral.

For an IN pipe, the field is incremented after each byte received from the peripheral and decremented after each byte read by the user from the pipe.

This field may be updated 1 clock cycle after the RWALL bit changes, so the user should not poll this field as an interrupt bit.

Downloaded from Arrow.com.

37.6.47 Host Pipe x Status Register (Isochronous Pipes)

Name: USBHS_HSTPIPISRx [x=0..9] (ISOPIPES)

Address: 0x40038530

Access: Read-only

31	30	29	28	27	26	25	24
–	PBYCT						
23	22	21	20	19	18	17	16
PBYCT				–	CFGOK	–	RWALL
15	14	13	12	11	10	9	8
CURRBK		NBUSYBK		–	–	DTSEQ	
7	6	5	4	3	2	1	0
SHORTPACKETI	CRCERRI	OVERFI	NAKEDI	PERRI	UNDERFI	TXOUTI	RXINI

This register view is relevant only if PTYPE = 0x1 in “Host Pipe x Configuration Register” on page 661.

- **RXINI: Received IN Data Interrupt**

0: Cleared when USBHS_HSTPIPICR.RXINIC = 1.

1: Set when a new USB message is stored in the current bank of the pipe. This triggers an interrupt if USBHS_HSTPIPIMR.RXINE = 1.

- **TXOUTI: Transmitted OUT Data Interrupt**

0: Cleared when USBHS_HSTPIPICR.TXOUTIC = 1.

1: Set when the current OUT bank is free and can be filled. This triggers an interrupt if USBHS_HSTPIPIMR.TXOUTE = 1.

- **UNDERFI: Underflow Interrupt**

This bit is set, for an isochronous and interrupt IN/OUT pipe, when an error flow occurs. This triggers an interrupt if the UNDERFIE bit = 1.

This bit is set, for an isochronous or interrupt OUT pipe, when a transaction underflow occurs in the current pipe (the pipe cannot send the OUT data packet in time because the current bank is not ready). A zero-length-packet (ZLP) is sent instead.

This bit is set, for an isochronous or interrupt IN pipe, when a transaction flow error occurs in the current pipe, i.e, the current bank of the pipe is not free while a new IN USB packet is received. This packet is not stored in the bank. For an interrupt pipe, the overflowed packet is ACKed to comply with the USB standard.

This bit is cleared when USBHS_HSTPIPICR.UNDERFIEC = 1.

- **PERRI: Pipe Error Interrupt**

0: Cleared when the error source bit is cleared.

1: Set when an error occurs on the current bank of the pipe. This triggers an interrupt if the USBHS_HSTPIPIMR.PERRE bit is set. Refer to the USBHS_HSTPIPERRx register to determine the source of the error.

- **NAKEDI: NAKed Interrupt**

0: Cleared when USBHS_HSTPIPICR.NAKEDIC = 1.

1: Set when a NAK has been received on the current bank of the pipe. This triggers an interrupt if the USBHS_HSTPIPIMR.NAKEDE bit = 1.

Downloaded from Arrow.com.

- **OVERFI: Overflow Interrupt**

0: Cleared when USBHS_HSTPIPICR.OVERFIC = 1.

1: Set when the current pipe has received more data than the maximum length of the current pipe. An interrupt is triggered if the USBHS_HSTPIPIMR.OVERFIE bit = 1.

- **CRCERRI: CRC Error Interrupt**

0: Cleared when USBHS_HSTPIPICR.CRCERRIC = 1.

1: Set when a CRC error occurs on the current bank of the pipe. This triggers an interrupt if the USBHS_HSTPIPIMR.TXSTPE bit = 1.

- **SHORTPACKETI: Short Packet Interrupt**

0: Cleared when USBHS_HSTPIPICR.SHORTPACKETIC = 1.

1: Set when a short packet is received by the host controller (packet length inferior to the PSIZE programmed field).

- **DTSEQ: Data Toggle Sequence**

This field indicates the data PID of the current bank.

Value	Name	Description
0	DATA0	Data0 toggle sequence
1	DATA1	Data1 toggle sequence
2	–	Reserved
3	–	Reserved

For an OUT pipe, this field indicates the data toggle of the next packet that is to be sent.

For an IN pipe, this field indicates the data toggle of the received packet stored in the current bank.

- **NBUSYBK: Number of Busy Banks**

This field indicates the number of busy banks.

For an OUT pipe, this field indicates the number of busy banks, filled by the user, ready for an OUT transfer. When all banks are busy, this triggers a PEP_x interrupt if USBHS_HSTPIPIMRx.NBUSYBKE = 1.

For an IN pipe, this field indicates the number of busy banks filled by IN transaction from the device. When all banks are free, this triggers a PEP_x interrupt if USBHS_HSTPIPIMRx.NBUSYBKE = 1.

Value	Name	Description
0	0_BUSY	0 busy bank (all banks free)
1	1_BUSY	1 busy bank
2	2_BUSY	2 busy banks
3	3_BUSY	3 busy banks

Downloaded from Arrow.com.

- **CURRBK: Current Bank**

For a non-control pipe, this field indicates the number of the current bank.

Value	Name	Description
0	BANK0	Current bank is bank0
1	BANK1	Current bank is bank1
2	BANK2	Current bank is bank2
3	–	Reserved

This field may be updated 1 clock cycle after the RWALL bit changes, so the user should not poll it as an interrupt bit.

- **RWALL: Read/Write Allowed**

For an OUT pipe, this bit is set when the current bank is not full, i.e., the software can write further data into the FIFO.

For an IN pipe, this bit is set when the current bank is not empty, i.e., the software can read further data from the FIFO.

This bit is cleared otherwise.

This bit is also cleared when the RXSTALLDI or the PERRI bit = 1.

- **CFGOK: Configuration OK Status**

This bit is set/cleared when the USBHS_HSTPIPCFGx.ALLOC bit is set.

This bit is set if the pipe x number of banks (USBHS_HSTPIPCFGx.PBK) and size (USBHS_HSTPIPCFGx.PSIZE) are correct compared to the maximal allowed number of banks and size for this pipe and to the maximal FIFO size (i.e., the DPRAM size).

If this bit is cleared, the user should rewrite correct values for the PBK and PSIZE fields in the USBHS_HSTPIPCFGx register.

- **PBYCT: Pipe Byte Count**

This field contains the byte count of the FIFO.

For an OUT pipe, the field is incremented after each byte written by the user into the pipe and decremented after each byte sent to the peripheral.

For an IN pipe, the field is incremented after each byte received from the peripheral and decremented after each byte read by the user from the pipe.

This field may be updated 1 clock cycle after the RWALL bit changes, so the user should not poll this field as an interrupt bit.

Downloaded from Arrow.com.

37.6.48 Host Pipe x Clear Register (Control, Bulk Pipes)

Name: USBHS_HSTPIPICRx [x=0..9]

Address: 0x40038560

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
SHORT PACKETIC	RXSTALLDIC	OVERFIC	NAKEDIC	–	TXSTPIC	TXOUTIC	RXINIC

This register view is relevant only if PTYPE = 0x0 or 0x2 in “Host Pipe x Configuration Register” on page 661.

For additional information, see “Host Pipe x Status Register (Control, Bulk Pipes)” on page 667.

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Clears the corresponding bit in USBHS_HSTPIPISRx.

- **RXINIC: Received IN Data Interrupt Clear**
- **TXOUTIC: Transmitted OUT Data Interrupt Clear**
- **TXSTPIC: Transmitted SETUP Interrupt Clear**
- **NAKEDIC: NAKed Interrupt Clear**
- **OVERFIC: Overflow Interrupt Clear**
- **RXSTALLDIC: Received STALLed Interrupt Clear**
- **SHORTPACKETIC: Short Packet Interrupt Clear**

Downloaded from Arrow.com.

37.6.49 Host Pipe x Clear Register (Interrupt Pipes)

Name: USBHS_HSTPIPICRx [x=0..9] (INTPIPES)

Address: 0x40038560

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
SHORT PACKETIC	RXSTALLDIC	OVERFIC	NAKEDIC	–	UNDERFIC	TXOUTIC	RXINIC

This register view is relevant only if PTYPE = 0x3 in “Host Pipe x Configuration Register” on page 661.

For additional information, see “Host Pipe x Status Register (Interrupt Pipes)” on page 670.

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Clears the corresponding bit in USBHS_HSTPIPISRx.

- **RXINIC: Received IN Data Interrupt Clear**
- **TXOUTIC: Transmitted OUT Data Interrupt Clear**
- **UNDERFIC: Underflow Interrupt Clear**
- **NAKEDIC: NAKed Interrupt Clear**
- **OVERFIC: Overflow Interrupt Clear**
- **RXSTALLDIC: Received STALLed Interrupt Clear**
- **SHORTPACKETIC: Short Packet Interrupt Clear**

Downloaded from Arrow.com.

37.6.50 Host Pipe x Clear Register (Isochronous Pipes)

Name: USBHS_HSTPIPICRx [x=0..9] (ISOPIPES)

Address: 0x40038560

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
SHORT PACKETIC	CRCERRIC	OVERFIC	NAKEDIC	–	UNDERFIC	TXOUTIC	RXINIC

This register view is relevant only if PTYPE = 0x1 in “Host Pipe x Configuration Register” on page 661.

For additional information, see “Host Pipe x Status Register (Isochronous Pipes)” on page 673.

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Clears the corresponding bit in USBHS_HSTPIPISRx.

- **RXINIC: Received IN Data Interrupt Clear**
- **TXOUTIC: Transmitted OUT Data Interrupt Clear**
- **UNDERFIC: Underflow Interrupt Clear**
- **NAKEDIC: NAKed Interrupt Clear**
- **OVERFIC: Overflow Interrupt Clear**
- **CRCERRIC: CRC Error Interrupt Clear**
- **SHORTPACKETIC: Short Packet Interrupt Clear**

Downloaded from Arrow.com.

37.6.51 Host Pipe x Set Register (Control, Bulk Pipes)

Name: USBHS_HSTPIPIFRx [x=0..9]

Address: 0x40038590

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	NBUSYBKS	–	–	–	–
7	6	5	4	3	2	1	0
SHORT PACKETIS	RXSTALLDIS	OVERFIS	NAKEDIS	PERRIS	TXSTPIS	TXOUTIS	RXINIS

This register view is relevant only if PTYPE = 0x0 or 0x2 in “Host Pipe x Configuration Register” on page 661.

For additional information, see “Host Pipe x Status Register (Control, Bulk Pipes)” on page 667.

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Sets the corresponding bit in USBHS_HSTPIPISRx, which may be useful for test or debug purposes.

- **RXINIS: Received IN Data Interrupt Set**
- **TXOUTIS: Transmitted OUT Data Interrupt Set**
- **TXSTPIS: Transmitted SETUP Interrupt Set**
- **PERRIS: Pipe Error Interrupt Set**
- **NAKEDIS: NAKed Interrupt Set**
- **OVERFIS: Overflow Interrupt Set**
- **RXSTALLDIS: Received STALLed Interrupt Set**
- **SHORTPACKETIS: Short Packet Interrupt Set**
- **NBUSYBKS: Number of Busy Banks Set**

Downloaded from Arrow.com.

37.6.52 Host Pipe x Set Register (Interrupt Pipes)

Name: USBHS_HSTPIPIFRx [x=0..9] (INTPIPES)

Address: 0x40038590

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	NBUSYBKS	–	–	–	–
7	6	5	4	3	2	1	0
SHORT PACKETIS	RXSTALLDIS	OVERFIS	NAKEDIS	PERRIS	UNDERFIS	TXOUTIS	RXINIS

This register view is relevant only if PTYPE = 0x3 in “Host Pipe x Configuration Register” on page 661.

For additional information, see “Host Pipe x Status Register (Interrupt Pipes)” on page 670.

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Sets the corresponding bit in USBHS_HSTPIPISRx, which may be useful for test or debug purposes.

- **RXINIS: Received IN Data Interrupt Set**
- **TXOUTIS: Transmitted OUT Data Interrupt Set**
- **UNDERFIS: Underflow Interrupt Set**
- **PERRIS: Pipe Error Interrupt Set**
- **NAKEDIS: NAKed Interrupt Set**
- **OVERFIS: Overflow Interrupt Set**
- **RXSTALLDIS: Received STALLed Interrupt Set**
- **SHORTPACKETIS: Short Packet Interrupt Set**
- **NBUSYBKS: Number of Busy Banks Set**

Downloaded from Arrow.com.

37.6.53 Host Pipe x Set Register (Isochronous Pipes)

Name: USBHS_HSTPIPIFRx [x=0..9] (ISOPIPES)

Address: 0x40038590

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	NBUSYBKS	–	–	–	–
7	6	5	4	3	2	1	0
SHORT PACKETIS	CRCERRIS	OVERFIS	NAKEDIS	PERRIS	UNDERFIS	TXOUTIS	RXINIS

This register view is relevant only if PTYPE = 0x1 in “Host Pipe x Configuration Register” on page 661.

For additional information, see “Host Pipe x Status Register (Isochronous Pipes)” on page 673.

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Sets the corresponding bit in USBHS_HSTPIPISRx, which may be useful for test or debug purposes.

- **RXINIS: Received IN Data Interrupt Set**
- **TXOUTIS: Transmitted OUT Data Interrupt Set**
- **UNDERFIS: Underflow Interrupt Set**
- **PERRIS: Pipe Error Interrupt Set**
- **NAKEDIS: NAKed Interrupt Set**
- **OVERFIS: Overflow Interrupt Set**
- **CRCERRIS: CRC Error Interrupt Set**
- **SHORTPACKETIS: Short Packet Interrupt Set**
- **NBUSYBKS: Number of Busy Banks Set**

Downloaded from Arrow.com.

37.6.54 Host Pipe x Mask Register (Control, Bulk Pipes)

Name: USBHS_HSTPIPIMRx [x=0..9]

Address: 0x400385C0

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	RSTDT	PFREEZE	PDISHDMA
15	14	13	12	11	10	9	8
–	FIFOCON	–	NBUSYBKE	–	–	–	–
7	6	5	4	3	2	1	0
SHORT PACKETIE	RXSTALLDE	OVERFIE	NAKEDE	PERRE	TXSTPE	TXOUTE	RXINE

This register view is relevant only if PTYPE = 0x0 or 0x2 in “Host Pipe x Configuration Register” on page 661.

- **RXINE: Received IN Data Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.RXINEC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.RXINE).

1: Set when USBHS_HSTPIPIER.RXINES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.RXINE).

- **TXOUTE: Transmitted OUT Data Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.TXOUTEC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.TXOUTE).

1: Set when USBHS_HSTPIPIER.TXOUTES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.TXOUTE).

- **TXSTPE: Transmitted SETUP Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.TXSTPEC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.TXSTPE).

1: Set when USBHS_HSTPIPIER.TXSTPES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.TXSTPE).

- **PERRE: Pipe Error Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.PERREC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.PERRE).

1: Set when USBHS_HSTPIPIER.PERRES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.PERRE).

- **NAKEDE: NAKed Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.NAKEDEC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.NAKEDE).

1: Set when USBHS_HSTPIPIER.NAKEDES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.NAKEDE).

Downloaded from Arrow.com.

- **OVERFIE: Overflow Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.OVERFIEC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.OVERFIE).

1: Set when USBHS_HSTPIPIER.OVERFIES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.OVERFIE).

- **RXSTALLDE: Received STALLed Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.RXSTALLDEC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.RXSTALLDE).

1: Set when USBHS_HSTPIPIER.RXSTALLDES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.RXSTALLDE).

- **SHORTPACKETIE: Short Packet Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.SHORTPACKETEC = 1. This disables the Transmitted IN Data IT (USBHS_HSTPIPIMR.SHORTPACKETE).

1: Set when USBHS_HSTPIPIER.SHORTPACKETIES = 1. This enables the Transmitted IN Data IT (USBHS_HSTPIPIMR.SHORTPACKETIE).

If this bit is set for non-control OUT pipes, a short packet transmission is guaranteed upon ending a DMA transfer, thus signaling an end of transfer, provided that End of DMA Buffer Output Enable (USBHS_HSTDMACONTROL.END_B_EN) and Automatic Switch (USBHS_HSTPIPCFG.AUTOSW) = 1.

- **NBUSYBKE: Number of Busy Banks Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.NBUSYBKEC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.NBUSYBKE).

1: Set when USBHS_HSTPIPIER.NBUSYBKES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.NBUSYBKE).

- **FIFOCON: FIFO Control**

For OUT and SETUP pipes:

0: Cleared when USBHS_HSTPIPIDR.FIFOCONC = 1. This sends the FIFO data and switches the bank.

1: Set when the current bank is free, at the same time as USBHS_HSTPIPISR.TXOUTI or TXSTPI.

For an IN pipe:

0: Cleared when USBHS_HSTPIPIDR.FIFOCONC = 1. This frees the current bank and switches to the next bank.

1: Set when a new IN message is stored in the current bank, at the same time as USBHS_HSTPIPISR.RXINI.

- **PDISHDMA: Pipe Interrupts Disable HDMA Request Enable**

See the USBHS_DEVEPTIMR.EPDISHDMA bit description.

Downloaded from Arrow.com.

- **PFREEZE: Pipe Freeze**

0: Cleared when USBHS_HSTPIPIDR.PFREEZEC = 1. This enables the pipe request generation.

1: Set when one of the following conditions is met:

- USBHS_HSTPIPIER.PFREEZES=
- The pipe is not configured.
- A STALL handshake has been received on the pipe.
- An error has occurred on the pipe (USBHS_HSTPIPISR.PERRI = 1).
- (INRQ+1) In requests have been processed.
- A Pipe Reset (USBHS_HSTPIP.PRSTx rising) has occurred.
- A Pipe Enable (USBHS_HSTPIP.PEN rising) has occurred.

This freezes the pipe request generation.

- **RSTDT: Reset Data Toggle**

0: No reset of the Data Toggle is ongoing.

0: Set when USBHS_HSTPIPIER.RSTDTS = 1. This resets the Data Toggle to its initial value for the current pipe.

Downloaded from Arrow.com.

37.6.55 Host Pipe x Mask Register (Interrupt Pipes)

Name: USBHS_HSTPIPIMRx [x=0..9] (INTPIPES)

Address: 0x400385C0

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	RSTDT	PFREEZE	PDISHDMA
15	14	13	12	11	10	9	8
–	FIFOCON	–	NBUSYBKE	–	–	–	–
7	6	5	4	3	2	1	0
SHORT PACKETIE	RXSTALLDE	OVERFIE	NAKEDE	PERRE	UNDERFIE	TXOUTE	RXINE

This register view is relevant only if PTYPE = 0x3 in “Host Pipe x Configuration Register” on page 661.

- **RXINE: Received IN Data Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.RXINEC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.RXINE).

1: Set when USBHS_HSTPIPIER.RXINES= 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.RXINE).

- **TXOUTE: Transmitted OUT Data Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.TXOUTEC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.TXOUTE).

1: Set when USBHS_HSTPIPIER.TXOUTES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.TXOUTE).

- **UNDERFIE: Underflow Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.UNDERFIEC= 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.UNDERFIE).

1: Set when USBHS_HSTPIPIER.UNDERFIES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.UNDERFIE).

- **PERRE: Pipe Error Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.PERREC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.PERRE).

1: Set when USBHS_HSTPIPIER.PERRES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.PERRE).

- **NAKEDE: NAKed Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.NAKEDEC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.NAKEDE).

1: Set when USBHS_HSTPIPIER.NAKEDES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.NAKEDE).

Downloaded from Arrow.com.

- **OVERFIE: Overflow Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.OVERFIEC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.OVERFIE).

1: Set when USBHS_HSTPIPIER.OVERFIES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.OVERFIE).

- **RXSTALLDE: Received STALLed Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.RXSTALLDEC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.RXSTALLDE).

1: Set when USBHS_HSTPIPIER.RXSTALLDES= 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.RXSTALLDE).

- **SHORTPACKETIE: Short Packet Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.SHORTPACKETEC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.SHORTPACKETE).

1: Set when USBHS_HSTPIPIER.SHORTPACKETIES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.SHORTPACKETIE).

If this bit is set for non-control OUT pipes, a short packet transmission is guaranteed upon ending a DMA transfer, thus signaling an end of transfer, provided that the End of DMA Buffer Output Enable (USBHS_HSTDMACONTROL.END_B_EN) bit and the Automatic Switch (USBHS_HSTPIPCFG.AUTOSW) bit = 1.

- **NBUSYBKE: Number of Busy Banks Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.NBUSYBKEC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.NBUSYBKE).

1: Set when USBHS_HSTPIPIER.NBUSYBKES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.NBUSYBKE).

- **FIFOCON: FIFO Control**

For OUT and SETUP pipes:

0: Cleared when USBHS_HSTPIPIDR.FIFOCONC = 1. This sends the FIFO data and switches the bank.

1: Set when the current bank is free, at the same time as USBHS_HSTPIPISR.TXOUTI or TXSTPI.

For IN pipes:

0: Cleared when USBHS_HSTPIPIDR.FIFOCONC = 1. This frees the current bank and switches to the next bank.

1: Set when a new IN message is stored in the current bank, at the same time as USBHS_HSTPIPISR.RXINI.

- **PDISHDMA: Pipe Interrupts Disable HDMA Request Enable**

See the USBHS_DEVEPTIMR.EPDISHDMA bit description.

Downloaded from Arrow.com.

- **PFREEZE: Pipe Freeze**

0: Cleared when USBHS_HSTPIPIDR.PFREEZEC = 1. This enables the pipe request generation.

1: Set when one of the following conditions is met:

- USBHS_HSTPIPIER.PFREEZES = 1
- The pipe is not configured.
- A STALL handshake has been received on the pipe.
- An error has occurred on the pipe (USBHS_HSTPIPISR.PERRI = 1).
- (INRQ+1) in requests have been processed.
- A Pipe Reset (USBHS_HSTPIP.PRSTx rising) has occurred.
- A Pipe Enable (USBHS_HSTPIP.PEN rising) has occurred.

This freezes the pipe request generation.

- **RSTDT: Reset Data Toggle**

0: 0: No reset of the Data Toggle is ongoing.

1: Set when USBHS_HSTPIPIER.RSTDTS = 1. This resets the Data Toggle to its initial value for the current pipe.

Downloaded from Arrow.com.

37.6.56 Host Pipe x Mask Register (Isochronous Pipes)

Name: USBHS_HSTPIPIMRx [x=0..9] (ISOPIPES)

Address: 0x400385C0

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	RSTDT	PFREEZE	PDISHDMA
15	14	13	12	11	10	9	8
–	FIFOCON	–	NBUSYBKE	–	–	–	–
7	6	5	4	3	2	1	0
SHORT PACKETIE	CRCERRE	OVERFIE	NAKEDE	PERRE	UNDERFIE	TXOUTE	RXINE

This register view is relevant only if PTYPE = 0x1 in “Host Pipe x Configuration Register” on page 661.

- **RXINE: Received IN Data Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.RXINEC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.RXINE).

1: Set when USBHS_HSTPIPIER.RXINES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.RXINE).

- **TXOUTE: Transmitted OUT Data Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.TXOUTEC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.TXOUTE).

1: Set when USBHS_HSTPIPIER.TXOUTES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.TXOUTE).

- **UNDERFIE: Underflow Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.UNDERFIEC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.UNDERFIE).

1: Set when USBHS_HSTPIPIER.UNDERFIES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.UNDERFIE).

- **PERRE: Pipe Error Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.PERREC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.PERRE).

1: Set when USBHS_HSTPIPIER.PERRES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.PERRE).

- **NAKEDE: NAKed Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.NAKEDEC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.NAKEDE).

1: Set when USBHS_HSTPIPIER.NAKEDES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.NAKEDE).

Downloaded from Arrow.com.

- **OVERFIE: Overflow Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.OVERFIEC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.OVERFIE).

1: Set when USBHS_HSTPIPIER.OVERFIES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.OVERFIE).

- **CRCERRE: CRC Error Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.CRCERREC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.CRCERRE).

1: Set when USBHS_HSTPIPIER.CRCERRES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.CRCERRE).

- **SHORTPACKETIE: Short Packet Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.SHORTPACKETEC = 1. This disables the Transmitted interrupt Data IT (USBHS_HSTPIPIMR.SHORTPACKETE).

1: Set when USBHS_HSTPIPIER.SHORTPACKETIES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.SHORTPACKETIE).

If this bit is set for non-control OUT pipes, a short packet transmission is guaranteed upon ending a DMA transfer, thus signaling an end of transfer, provided that the End of DMA Buffer Output Enable (USBHS_HSTDMACONTROL.END_B_EN) bit and the Automatic Switch (USBHS_HSTPIPCFG.AUTOSW) bit = 1.

- **NBUSYBKE: Number of Busy Banks Interrupt Enable**

0: Cleared when USBHS_HSTPIPIDR.NBUSYBKEC = 1. This disables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.NBUSYBKE).

1: Set when USBHS_HSTPIPIER.NBUSYBKES = 1. This enables the Transmitted IN Data interrupt (USBHS_HSTPIPIMR.NBUSYBKE).

- **FIFOCON: FIFO Control**

For OUT and SETUP pipes:

0: Cleared when USBHS_HSTPIPIDR.FIFOCONC = 1. This sends the FIFO data and switches the bank.

1: Set when the current bank is free, at the same time as USBHS_HSTPIPISR.TXOUTI or TXSTPI.

For IN pipes:

0: Cleared when USBHS_HSTPIPIDR.FIFOCONC = 1. This frees the current bank and switches to the next bank.

1: Set when a new IN message is stored in the current bank, at the same time as USBHS_HSTPIPISR.RXINI.

- **PDISHDMA: Pipe Interrupts Disable HDMA Request Enable**

See the USBHS_DEVEPTIMR.EPDISHDMA bit description.

Downloaded from Arrow.com.

- **PFREEZE: Pipe Freeze**

0: Cleared when USBHS_HSTPIPIDR.PFREEZEC = 1. This enables the pipe request generation.

1: Set when one of the following conditions is met:

- USBHS_HSTPIPIER.PFREEZES = 1.
- The pipe is not configured.
- A STALL handshake has been received on the pipe.
- An error has occurred on the pipe (USBHS_HSTPIPISR.PERRI = 1).
- (INRQ+1) In requests have been processed.
- A Pipe Reset (USBHS_HSTPIP.PRSTx rising) has occurred.
- A Pipe Enable (USBHS_HSTPIP.PEN rising) has occurred.

This freezes the pipe request generation.

- **RSTDT: Reset Data Toggle**

0: No reset of the Data Toggle is ongoing.

1: Set when USBHS_HSTPIPIER.RSTDTS = 1. This resets the Data Toggle to its initial value for the current pipe.

Downloaded from Arrow.com.

37.6.57 Host Pipe x Disable Register (Control, Bulk Pipes)

Name: USBHS_HSTPIPIDRx [x=0..9]

Address: 0x40038620

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	PFREEZEC	PDISHDMAC
15	14	13	12	11	10	9	8
–	FIFOCONC	–	NBUSYBKEC	–	–	–	–
7	6	5	4	3	2	1	0
SHORT PACKETIEC	RXSTALLDEC	OVERFIEC	NAKEDEC	PERREC	TXSTPEC	TXOUTEC	RXINEC

This register view is relevant only if PTYPE = 0x0 or 0x2 in “Host Pipe x Configuration Register” on page 661.

For additional information, see “Host Pipe x Mask Register (Control, Bulk Pipes)” on page 682.

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Clears the corresponding bit in USBHS_HSTPIPIMRx.

- **RXINEC: Received IN Data Interrupt Disable**
- **TXOUTEC: Transmitted OUT Data Interrupt Disable**
- **TXSTPEC: Transmitted SETUP Interrupt Disable**
- **PERREC: Pipe Error Interrupt Disable**
- **NAKEDEC: NAKed Interrupt Disable**
- **OVERFIEC: Overflow Interrupt Disable**
- **RXSTALLDEC: Received STALLed Interrupt Disable**
- **SHORTPACKETIEC: Short Packet Interrupt Disable**
- **NBUSYBKEC: Number of Busy Banks Disable**
- **FIFOCONC: FIFO Control Disable**
- **PDISHDMAC: Pipe Interrupts Disable HDMA Request Disable**
- **PFREEZEC: Pipe Freeze Disable**

Downloaded from Arrow.com.

37.6.58 Host Pipe x Disable Register (Interrupt Pipes)

Name: USBHS_HSTPIPIDRx [x=0..9] (INTPIPES)

Address: 0x40038620

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	PFREEZEC	PDISHDMAC
15	14	13	12	11	10	9	8
–	FIFOCONC	–	NBUSYBKEC	–	–	–	–
7	6	5	4	3	2	1	0
SHORT PACKETIEC	RXSTALLDEC	OVERFIEC	NAKEDEC	PERREC	UNDERFIEC	TXOUTEC	RXINEC

This register view is relevant only if PTYPE = 0x3 in “Host Pipe x Configuration Register” on page 661.

For additional information, see “Host Pipe x Mask Register (Interrupt Pipes)” on page 685.

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Clears the corresponding bit in USBHS_HSTPIPIMRx.

- **RXINEC: Received IN Data Interrupt Disable**
- **TXOUTEC: Transmitted OUT Data Interrupt Disable**
- **UNDERFIEC: Underflow Interrupt Disable**
- **PERREC: Pipe Error Interrupt Disable**
- **NAKEDEC: NAKed Interrupt Disable**
- **OVERFIEC: Overflow Interrupt Disable**
- **RXSTALLDEC: Received STALLed Interrupt Disable**
- **SHORTPACKETIEC: Short Packet Interrupt Disable**
- **NBUSYBKEC: Number of Busy Banks Disable**
- **FIFOCONC: FIFO Control Disable**
- **PDISHDMAC: Pipe Interrupts Disable HDMA Request Disable**
- **PFREEZEC: Pipe Freeze Disable**

Downloaded from Arrow.com.

37.6.59 Host Pipe x Disable Register (Isochronous Pipes)

Name: USBHS_HSTPIPIDRx [x=0..9] (ISOPIPES)

Address: 0x40038620

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	PFREEZEC	PDISHDMAC
15	14	13	12	11	10	9	8
–	FIFOCONC	–	NBUSYBKEC	–	–	–	–
7	6	5	4	3	2	1	0
SHORT PACKETIEC	CRCERREC	OVERFIEC	NAKEDEC	PERREC	UNDERFIEC	TXOUTEC	RXINEC

This register view is relevant only if PTYPE = 0x1 in “Host Pipe x Configuration Register” on page 661.

For additional information, see “Host Pipe x Mask Register (Isochronous Pipes)” on page 688.

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Clears the corresponding bit in USBHS_HSTPIPIMRx.

- **RXINEC: Received IN Data Interrupt Disable**
- **TXOUTEC: Transmitted OUT Data Interrupt Disable**
- **UNDERFIEC: Underflow Interrupt Disable**
- **PERREC: Pipe Error Interrupt Disable**
- **NAKEDEC: NAKed Interrupt Disable**
- **OVERFIEC: Overflow Interrupt Disable**
- **CRCERREC: CRC Error Interrupt Disable**
- **SHORTPACKETIEC: Short Packet Interrupt Disable**
- **NBUSYBKEC: Number of Busy Banks Disable**
- **FIFOCONC: FIFO Control Disable**
- **PDISHDMAC: Pipe Interrupts Disable HDMA Request Disable**
- **PFREEZEC: Pipe Freeze Disable**

Downloaded from Arrow.com.

37.6.60 Host Pipe x Enable Register (Control, Bulk Pipes)

Name: USBHS_HSTPIPIERx [x=0..9]

Address: 0x400385F0

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	RSTDTS	PFREEZES	PDISHDMAS
15	14	13	12	11	10	9	8
–	–	–	NBUSYBKES	–	–	–	–
7	6	5	4	3	2	1	0
SHORT PACKETIES	RXSTALLDES	OVERFIES	NAKEDES	PERRES	TXSTPES	TXOUTES	RXINES

This register view is relevant only if PTYPE = 0x0 or 0x2 in “Host Pipe x Configuration Register” on page 661.

For additional information, see “Host Pipe x Mask Register (Control, Bulk Pipes)” on page 682.

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Sets the corresponding bit in USBHS_HSTPIPIMRx.

- **RXINES: Received IN Data Interrupt Enable**
- **TXOUTES: Transmitted OUT Data Interrupt Enable**
- **TXSTPES: Transmitted SETUP Interrupt Enable**
- **PERRES: Pipe Error Interrupt Enable**
- **NAKEDES: NAKed Interrupt Enable**
- **OVERFIES: Overflow Interrupt Enable**
- **RXSTALLDES: Received STALLed Interrupt Enable**
- **SHORTPACKETIES: Short Packet Interrupt Enable**
- **NBUSYBKES: Number of Busy Banks Enable**
- **PDISHDMAS: Pipe Interrupts Disable HDMA Request Enable**
- **PFREEZES: Pipe Freeze Enable**
- **RSTDTS: Reset Data Toggle Enable**

Downloaded from Arrow.com.

37.6.61 Host Pipe x Enable Register (Interrupt Pipes)

Name: USBHS_HSTPIPIERx [x=0..9] (INTPIPES)

Address: 0x400385F0

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	RSTDTS	PFREEZES	PDISHDMAS
15	14	13	12	11	10	9	8
–	–	–	NBUSYBKES	–	–	–	–
7	6	5	4	3	2	1	0
SHORT PACKETIES	RXSTALLDES	OVERFIES	NAKEDES	PERRES	UNDERFIES	TXOUTES	RXINES

This register view is relevant only if PTYPE = 0x3 in “Host Pipe x Configuration Register” on page 661.

For additional information, see “Host Pipe x Mask Register (Interrupt Pipes)” on page 685.

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Sets the corresponding bit in USBHS_HSTPIPIMRx.

- **RXINES: Received IN Data Interrupt Enable**
- **TXOUTES: Transmitted OUT Data Interrupt Enable**
- **UNDERFIES: Underflow Interrupt Enable**
- **PERRES: Pipe Error Interrupt Enable**
- **NAKEDES: NAKed Interrupt Enable**
- **OVERFIES: Overflow Interrupt Enable**
- **RXSTALLDES: Received STALLed Interrupt Enable**
- **SHORTPACKETIES: Short Packet Interrupt Enable**
- **NBUSYBKES: Number of Busy Banks Enable**
- **PDISHDMAS: Pipe Interrupts Disable HDMA Request Enable**
- **PFREEZES: Pipe Freeze Enable**
- **RSTDTS: Reset Data Toggle Enable**

Downloaded from Arrow.com.

37.6.62 Host Pipe x Enable Register (Isochronous Pipes)

Name: USBHS_HSTPIPIERx [x=0..9] (ISOPIPES)

Address: 0x400385F0

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	RSTDTS	PFREEZES	PDISHDMAS
15	14	13	12	11	10	9	8
–	–	–	NBUSYBKES	–	–	–	–
7	6	5	4	3	2	1	0
SHORT PACKETIES	CRCERRES	OVERFIES	NAKEDES	PERRES	UNDERFIES	TXOUTES	RXINES

This register view is relevant only if PTYPE = 0x1 in “Host Pipe x Configuration Register” on page 661.

For additional information, see “Host Pipe x Mask Register (Isochronous Pipes)” on page 688.

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Sets the corresponding bit in USBHS_HSTPIPIMRx.

- **RXINES: Received IN Data Interrupt Enable**
- **TXOUTES: Transmitted OUT Data Interrupt Enable**
- **UNDERFIES: Underflow Interrupt Enable**
- **PERRES: Pipe Error Interrupt Enable**
- **NAKEDES: NAKed Interrupt Enable**
- **OVERFIES: Overflow Interrupt Enable**
- **CRCERRES: CRC Error Interrupt Enable**
- **SHORTPACKETIES: Short Packet Interrupt Enable**
- **NBUSYBKES: Number of Busy Banks Enable**
- **PDISHDMAS: Pipe Interrupts Disable HDMA Request Enable**
- **PFREEZES: Pipe Freeze Enable**
- **RSTDTS: Reset Data Toggle Enable**

Downloaded from Arrow.com.

37.6.63 Host Pipe x IN Request Register

Name: USBHS_HSTPIPINRQx [x=0..9]

Address: 0x40038650

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	INMODE
7	6	5	4	3	2	1	0
INRQ							

- **INRQ: IN Request Number before Freeze**

This field contains the number of IN transactions before the USBHS freezes the pipe. The USBHS performs (INRQ+1) IN requests before freezing the pipe. This counter is automatically decreased by 1 each time an IN request has been successfully performed.

This register has no effect when INMODE = 1.

- **INMODE: IN Request Mode**

0: Performs a pre-defined number of IN requests. This number is the INRQ field.

1: Enables the USBHS to perform infinite IN requests when the pipe is not frozen.

Downloaded from Arrow.com.

37.6.64 Host Pipe x Error Register

Name: USBHS_HSTPIPERRx [x=0..9]

Address: 0x40038680

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	COUNTER		CRC16	TIMEOUT	PID	DATAPID	DATATGL

Writing a zero in a bit/field in this register clears the bit/field. Writing a one has no effect.

- **DATATGL: Data Toggle Error**

0: No Data Toggle error occurred since last clear of this bit.

1: This bit is automatically set when a Data Toggle error has been detected.

- **DATAPID: Data PID Error**

0: No Data PID error occurred since last clear of this bit.

1: This bit is automatically set when a Data PID error has been detected.

- **PID: PID Error**

0: No PID error occurred since last clear of this bit.

1: This bit is automatically set when a PID error has been detected.

- **TIMEOUT: Time-Out Error**

0: No Time-Out error occurred since last clear of this bit.

1: This bit is automatically set when a Time-Out error has been detected.

- **CRC16: CRC16 Error**

0: No CRC16 error occurred since last clear of this bit.

1: This bit is automatically set when a CRC16 error has been detected.

- **COUNTER: Error Counter**

This field is incremented each time an error occurs (CRC16, TIMEOUT, PID, DATAPID or DATATGL).

This field is cleared when receiving a USB packet free of error.

When this field reaches 3 (i.e., 3 consecutive errors), this pipe is automatically frozen (USBHS_HSTPIPIMRx.PFREEZE is set).

Downloaded from Arrow.com.

37.6.65 Host DMA Channel x Next Descriptor Address Register

Name: USBHS_HSTDMANXTDSCx [x=1..7]

Address: 0x40038710 [1], 0x40038720 [2], 0x40038730 [3], 0x40038740 [4], 0x40038750 [5], 0x40038760 [6], 0x40038770 [7]

Access: Read/Write

31	30	29	28	27	26	25	24
NXT_DSC_ADD							
23	22	21	20	19	18	17	16
NXT_DSC_ADD							
15	14	13	12	11	10	9	8
NXT_DSC_ADD							
7	6	5	4	3	2	1	0
NXT_DSC_ADD							

- **NXT_DSC_ADD: Next Descriptor Address**

This field points to the next channel descriptor to be processed. This channel descriptor must be aligned, so bits 0 to 3 of the address must be equal to zero.

Downloaded from Arrow.com.

37.6.66 Host DMA Channel x Address Register

Name: USBHS_HSTDMAADDRESSx [x=1..7]

Address: 0x40038714 [1], 0x40038724 [2], 0x40038734 [3], 0x40038744 [4], 0x40038754 [5], 0x40038764 [6], 0x40038774 [7]

Access: Read/Write

31	30	29	28	27	26	25	24
BUFF_ADD							
23	22	21	20	19	18	17	16
BUFF_ADD							
15	14	13	12	11	10	9	8
BUFF_ADD							
7	6	5	4	3	2	1	0
BUFF_ADD							

- **BUFF_ADD: Buffer Address**

This field determines the AHB bus starting address of a DMA channel transfer.

Channel start and end addresses may be aligned on any byte boundary.

The firmware can write this field only when the USBHS_HSTDMASTATUS.CHANN_ENB bit is cleared.

This field is updated at the end of the address phase of the current access to the AHB bus. It is incremented by the access byte width. The access width is 4 bytes (or less) at packet start or end, if the start or end address is not aligned on a word boundary.

The packet start address is either the channel start address or the next channel address to be accessed in the channel buffer.

The packet end address is either the channel end address or the latest channel address accessed in the channel buffer.

The channel start address is written by software or loaded from the descriptor. The channel end address is either determined by the end of buffer or the USB device, or by the USB end of transfer if the USBHS_HSTDMACONTROLx.END_TR_EN bit is set.

Downloaded from Arrow.com.

37.6.67 Host DMA Channel x Control Register

Name: USBHS_HSTDMACONTROLx [x=1..7]

Address: 0x40038718 [1], 0x40038728 [2], 0x40038738 [3], 0x40038748 [4], 0x40038758 [5], 0x40038768 [6], 0x40038778 [7]

Access: Read/Write

31	30	29	28	27	26	25	24
BUFF_LENGTH							
23	22	21	20	19	18	17	16
BUFF_LENGTH							
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
BURST_LCK	DESC_LD_IT	END_BUFFIT	END_TR_IT	END_B_EN	END_TR_EN	LDNXT_DSC	CHANN_ENB

- **CHANN_ENB: Channel Enable Command**

0: The DMA channel is disabled and no transfer occurs upon request. This bit is also cleared by hardware when the channel source bus is disabled at the end of the buffer.

If the LDNXT_DSC bit has been cleared by descriptor loading, the firmware has to set the corresponding CHANN_ENB bit to start the described transfer, if needed.

If the LDNXT_DSC bit is cleared, the channel is frozen and the channel registers may then be read and/or written reliably as soon as both the USBHS_HSTDMASTATUS.CHANN_ENB and the CHANN_ACT flags read as 0.

If a channel request is currently serviced when this bit is cleared, the DMA FIFO buffer is drained until it is empty, then the USBHS_HSTDMASTATUS.CHANN_ENB bit is cleared.

If the LDNXT_DSC bit is set or after it has been cleared, the currently loaded descriptor is skipped (no data transfer occurs) and the next descriptor is immediately loaded.

1: The USBHS_HSTDMASTATUS.CHANN_ENB bit is set, enabling DMA channel data transfer. Then, any pending request starts the transfer. This may be used to start or resume any requested transfer.

- **LDNXT_DSC: Load Next Channel Transfer Descriptor Enable Command**

0: No channel register is loaded after the end of the channel transfer.

1: The channel controller loads the next descriptor after the end of the current transfer, i.e., when the USBHS_HSTDMASTATUS.CHANN_ENB bit is reset.

If the CHANN_ENB bit is cleared, the next descriptor is loaded immediately upon transfer request.

DMA Channel Control Command Summary:

Value LDNXT_DSC	Value CHANN_ENB	Name	Description
0	0	STOP_NOW	Stop now
0	1	RUN_AND_STOP	Run and stop at end of buffer
1	0	LOAD_NEXT_DESC	Load next descriptor now
1	1	RUN_AND_LINK	Run and link at end of buffer

Downloaded from Arrow.com.

- **END_TR_EN: End of Transfer Enable Control (OUT transfers only)**

0: USB end of transfer is ignored.

1: The USBHS device can put an end to the current buffer transfer.

When set, a BULK or INTERRUPT short packet closes the current buffer and the USBHS_HSTDMASTATUSx.END_TR_ST flag is raised.

This is intended for a USBHS non-prenegotiated USB transfer size.

- **END_B_EN: End of Buffer Enable Control**

0: DMA Buffer End has no impact on USB packet transfer.

1: The pipe can validate the packet (according to the values programmed in the USBHS_HSTPIPCFGx.AUTOSW and USBHS_HSTPIPIMRx.SHORTPACKETIE fields) at DMA Buffer End, i.e., when USBHS_HSTDMASTATUS.BUFF_COUNT reaches 0.

This is mainly for short packet OUT validations initiated by the DMA reaching the end of buffer, but could be used for IN packet truncation (discarding of unwanted packet data) at the end of DMA buffer.

- **END_TR_IT: End of Transfer Interrupt Enable**

0: Completion of a USBHS device-initiated buffer transfer does not trigger any interrupt at USBHS_HSTDMASTATUSx.END_TR_ST rising.

1: An interrupt is sent after the buffer transfer is complete, if the USBHS device has ended the buffer transfer.

Use when the receive size is unknown.

- **END_BUFFIT: End of Buffer Interrupt Enable**

0: USBHS_HSTDMASTATUSx.END_BF_ST rising does not trigger any interrupt.

1: An interrupt is generated when USBHS_HSTDMASTATUSx.BUFF_COUNT reaches zero.

- **DESC_LD_IT: Descriptor Loaded Interrupt Enable**

0: USBHS_HSTDMASTATUSx.DESC_LDST rising does not trigger any interrupt.

1: An interrupt is generated when a descriptor has been loaded from the bus.

- **BURST_LCK: Burst Lock Enable**

0: The DMA never locks the bus access.

1: USB packets AHB data bursts are locked for maximum optimization of the bus bandwidth usage and maximization of fly-by AHB burst duration.

- **BUFF_LENGTH: Buffer Byte Length (Write-only)**

This field determines the number of bytes to be transferred until end of buffer. The maximum channel transfer size (32 KBytes) is reached when this field is 0 (default value). If the transfer size is unknown, this field should be set to 0, but the transfer end may occur earlier under USB device control.

When this field is written, the USBHS_HSTDMASTATUSx.BUFF_COUNT field is updated with the write value.

Notes:
1. Bits [31:2] are only writable when issuing a channel Control Command other than “Stop Now”.
2. For reliability, it is highly recommended to wait for both the USBHS_HSTDMASTATUSx.CHAN_ACT and the CHAN_ENB flags to be at 0, thus ensuring the channel has been stopped before issuing a command other than “Stop Now”.

Downloaded from Arrow.com.

37.6.68 Host DMA Channel x Status Register

Name: USBHS_HSTDMASTATUSx [x=1..7]

Address: 0x4003871C [1], 0x4003872C [2], 0x4003873C [3], 0x4003874C [4], 0x4003875C [5], 0x4003876C [6], 0x4003877C [7]

Access: Read/Write

31	30	29	28	27	26	25	24
BUFF_COUNT							
23	22	21	20	19	18	17	16
BUFF_COUNT							
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	DESC_LDST	END_BF_ST	END_TR_ST	–	–	CHANN_ACT	CHANN_ENB

- **CHANN_ENB: Channel Enable Status**

0: If cleared, the DMA channel no longer transfers data, and may load the next descriptor if the USBHS_HSTDMACONTROLx.LDNXT_DSC bit is set.

When any transfer is ended either due to an elapsed byte count or to completion of a USBHS device-initiated transfer, this bit is automatically reset.

1: If set, the DMA channel is currently enabled and transfers data upon request.

This bit is normally set or cleared by writing into the USBHS_HSTDMACONTROLx.CHANN_ENB bit field either by software or descriptor loading.

If a channel request is currently serviced when the USBHS_HSTDMACONTROLx.CHANN_ENB bit is cleared, the DMA FIFO buffer is drained until it is empty, then this status bit is cleared.

- **CHANN_ACT: Channel Active Status**

0: The DMA channel is no longer trying to source the packet data.

When a packet transfer is ended, this bit is automatically reset.

1: The DMA channel is currently trying to source packet data, i.e., selected as the highest-priority requesting channel.

When a packet transfer cannot be completed due to an END_BF_ST, this flag stays set during the next channel descriptor load (if any) and potentially until completion of a USBHS packet transfer, if allowed by the new descriptor.

- **END_TR_ST: End of Channel Transfer Status**

0: Cleared automatically when read by software.

1: Set by hardware when the last packet transfer is complete, if the USBHS device has ended the transfer.

Valid until the CHANN_ENB flag is cleared at the end of the next buffer transfer.

- **END_BF_ST: End of Channel Buffer Status**

0: Cleared automatically when read by software.

1: Set by hardware when the BUFF_COUNT count-down reaches zero.

Valid until the CHANN_ENB flag is cleared at the end of the next buffer transfer.

Downloaded from Arrow.com.

- **DESC_LDST: Descriptor Loaded Status**

0: Cleared automatically when read by software.

1: Set by hardware when a descriptor has been loaded from the system bus.

Valid until the CHANN_ENB flag is cleared at the end of the next buffer transfer.

- **BUFF_COUNT: Buffer Byte Count**

This field determines the current number of bytes still to be transferred for this buffer.

This field is decremented from the AHB source bus access byte width at the end of this bus address phase.

The access byte width is 4 by default, or less, at DMA start or end, if the start or end address is not aligned on a word boundary.

At the end of buffer, the DMA accesses the USBHS device only for the number of bytes needed to complete it.

Note: For IN pipes, if the receive buffer byte length (USBHS_HSTDMACONTROL.BUFF_LENGTH) has been defaulted to zero because the USB transfer length is unknown, the actual buffer byte length received is 0x10000-BUFF_COUNT.

Downloaded from Arrow.com.

38. Ethernet MAC (GMAC)

38.1 Description

The Ethernet MAC (GMAC) module implements a 10/100 Mbps Ethernet MAC compatible with the IEEE 802.3 standard. The GMAC can operate in either half or full duplex mode at all supported speeds. The GMAC Network Configuration Register is used to select the speed, duplex mode and interface type (MII, RMII).

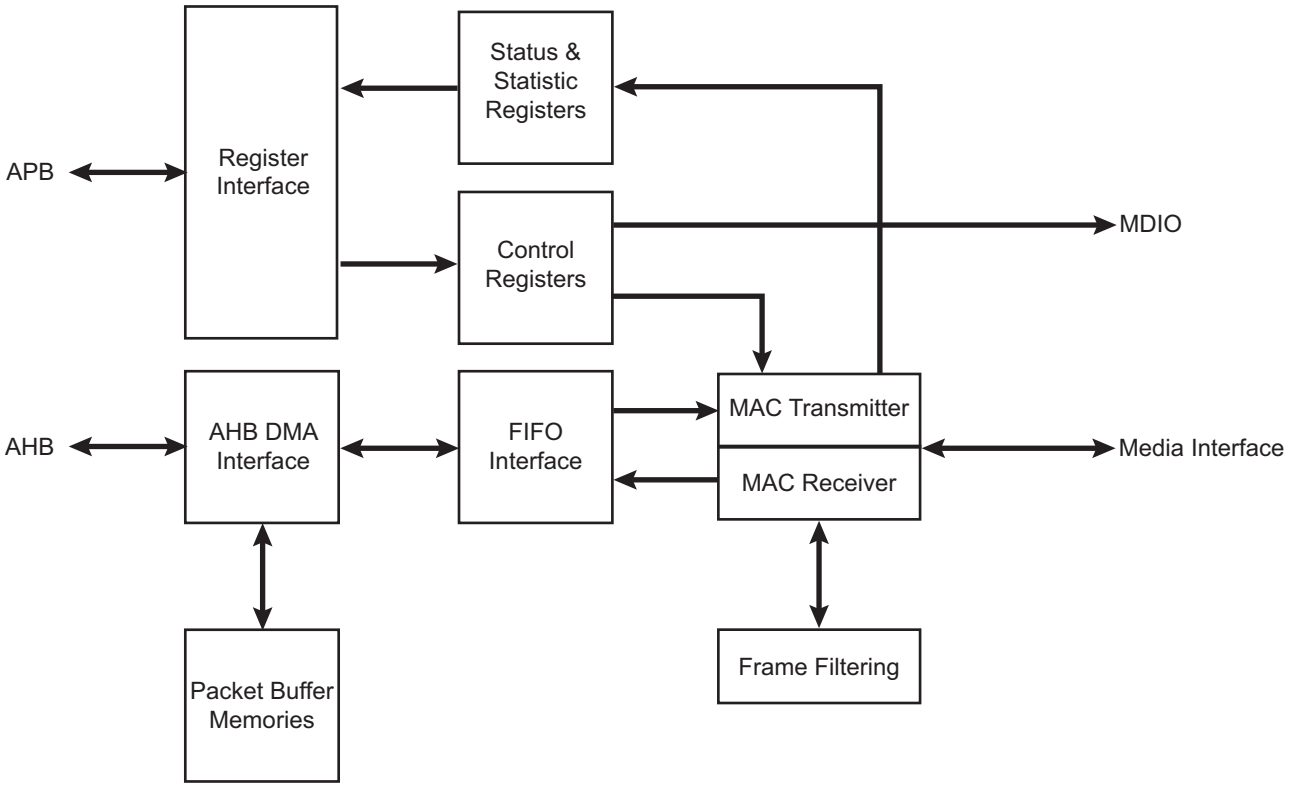
38.2 Embedded Characteristics

- Compatible with IEEE Standard 802.3
- 10, 100 Mbps Operation
- Full and Half Duplex Operation at all Supported Speeds of Operation
- Statistics Counter Registers for RMON/MIB
- MII/RMII Interface to the Physical Layer
- Integrated Physical Coding
- Direct Memory Access (DMA) Interface to External Memory
- Support for 3 Priority Queues in DMA
- 8 Kbytes Transmit RAM (2 KB for Queue 0, 2 KB for Queue 1, 4 KB for Queue 2) and 4 Kbytes Receive RAM
- Programmable Burst Length and Endianism for DMA
- Interrupt Generation to Signal Receive and Transmit Completion, Errors or Other Events
- Automatic Pad and Cyclic Redundancy Check (CRC) Generation on Transmitted Frames
- Automatic Discard of Frames Received with Errors
- Receive and Transmit IP, TCP and UDP Checksum Offload. Both IPv4 and IPv6 Packet Types Supported
- Address Checking Logic for Four Specific 48-bit Addresses, Four Type IDs, Promiscuous Mode, Hash Matching of Unicast and Multicast Destination Addresses and Wake-on-LAN
- Management Data Input/Output (MDIO) Interface for Physical Layer Management
- Support for Jumbo Frames up to 10240 Bytes
- Full Duplex Flow Control with Recognition of Incoming Pause Frames and Hardware Generation of Transmitted Pause Frames
- Half Duplex Flow Control by Forcing Collisions on Incoming Frames
- Support for 802.1Q VLAN Tagging with Recognition of Incoming VLAN and Priority Tagged Frames
- Support for 802.1Qbb Priority-based Flow Control
- Programmable Inter Packet Gap (IPG) Stretch
- Recognition of IEEE 1588 PTP Frames
- IEEE 1588 Time Stamp Unit (TSU)
- Support for 802.1AS Timing and Synchronization
- Supports 802.1Qav Traffic Shaping on Two Highest Priority Queues

Downloaded from Arrow.com.

38.3 Block Diagram

Figure 38-1. Block Diagram



38.4 Signal Interface

The GMAC includes the following signal interfaces:

- MII, RMII to an external PHY
- MDIO interface for external PHY management
- Slave APB interface for accessing GMAC registers
- Master AHB interface for memory access
- GTSUCOMP signal for TSU timer count value comparison

Table 38-1. GMAC Connections in Different Modes

Signal Name	Function	MII	RMII
GTXCK	Transmit Clock or Reference Clock	TXCK	REFCK
GTXEN	Transmit Enable	TXEN	TXEN
GTX[3..0]	Transmit Data	TXD[3:0]	TXD[1:0]
GTXER	Transmit Coding Error	TXER	Not Used
GRXCK	Receive Clock	RXCK	Not Used
GRXDV	Receive Data Valid	RXDV	CRSDV
GRX[3..0]	Receive Data	RXD[3:0]	RXD[1:0]
GRXER	Receive Error	RXER	RXER
GCRS	Carrier Sense and Data Valid	CRS	Not Used

Downloaded from Arrow.com.

Table 38-1. GMAC Connections in Different Modes (Continued)

Signal Name	Function	MII	RMII
GCOL	Collision Detect	COL	Not Used
GMDC	Management Data Clock	MDC	MDC
GMDIO	Management Data Input/Output	MDIO	MDIO

38.5 Product Dependencies

38.5.1 I/O Lines

The pins used for interfacing the GMAC may be multiplexed with PIO lines. The programmer must first program the PIO Controller to assign the pins to their peripheral function. If I/O lines of the GMAC are not used by the application, they can be used for other purposes by the PIO Controller.

Table 38-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
GMAC	GCOL	PD13	A
GMAC	GCRS	PD10	A
GMAC	GMDC	PD8	A
GMAC	GMDIO	PD9	A
GMAC	GRXCK	PD14	A
GMAC	GRXDV	PD4	A
GMAC	GRXER	PD7	A
GMAC	GRX0	PD5	A
GMAC	GRX1	PD6	A
GMAC	GRX2	PD11	A
GMAC	GRX3	PD12	A
GMAC	GTSUCOMP	PB1	B
GMAC	GTSUCOMP	PB12	B
GMAC	GTSUCOMP	PD11	C
GMAC	GTSUCOMP	PD20	C
GMAC	GTXCK	PD0	A
GMAC	GTXEN	PD1	A
GMAC	GTXER	PD17	A
GMAC	GTX0	PD2	A
GMAC	GTX1	PD3	A
GMAC	GTX2	PD15	A
GMAC	GTX3	PD16	A

38.5.2 Power Management

The GMAC is not continuously clocked. The user must first enable the GMAC clock in the Power Management Controller before using it.

Downloaded from Arrow.com.

38.5.3 Interrupt Sources

The GMAC interrupt line is connected to one of the internal sources of the interrupt controller. Using the GMAC interrupt requires prior programming of the interrupt controller.

The GMAC features three interrupt sources. Interrupt source 39 is the interrupt number for the GMAC priority queue 0 and all general interrupts. The interrupt sources 66 and 67 are the interrupt numbers for GMAC priority queue Q1 and Q2, respectively.

Table 38-3. Peripheral IDs

Instance	ID
GMAC	39

38.6 Functional Description

38.6.1 Media Access Controller

The Media Access Controller (MAC) transmit block takes data from FIFO, adds preamble and, if necessary, pad and frame check sequence (FCS). Both half duplex and full duplex Ethernet modes of operation are supported. When operating in half duplex mode, the MAC transmit block generates data according to the carrier sense multiple access with collision detect (CSMA/CD) protocol. The start of transmission is deferred if carrier sense (CRS) is active. If collision (COL) becomes active during transmission, a jam sequence is asserted and the transmission is retried after a random back off. The CRS and COL signals have no effect in full duplex mode.

The MAC receive block checks for valid preamble, FCS, alignment and length, and presents received frames to the MAC address checking block and FIFO. Software can configure the GMAC to receive jumbo frames up to 10240 bytes. It can optionally strip CRC from the received frame prior to transfer to FIFO.

The address checker recognizes four specific 48-bit addresses, can recognize four different type ID values, and contains a 64-bit Hash register for matching multicast and unicast addresses as required. It can recognize the broadcast address of all ones and copy all frames. The MAC can also reject all frames that are not VLAN tagged and recognize Wake on LAN events.

The MAC receive block supports offloading of IP, TCP and UDP checksum calculations (both IPv4 and IPv6 packet types supported), and can automatically discard bad checksum frames.

38.6.2 1588 Time Stamp Unit

The 1588 time stamp unit (TSU) is implemented as a 94-bit timer.

The 48 upper bits [93:46] of the timer count seconds and are accessible in the "GMAC 1588 Timer Seconds High Register" (GMAC_TSH) and "GMAC 1588 Timer Seconds Low Register" (GMAC_TSL). The 30 lower bits [45:16] of the timer count nanoseconds and are accessible in the "GMAC 1588 Timer Nanoseconds Register" (GMAC_TN). The lowest 16 bits [15:0] of the timer count sub-nanoseconds.

The 46 lower bits roll over when they have counted to one second. The timer increments by a programmable period (to approximately 15.2 femtoseconds resolution) with each MCK period and can also be adjusted in 1ns resolution (incremented or decremented) through APB register accesses.

38.6.3 AHB Direct Memory Access Interface

The GMAC DMA controller is connected to the MAC FIFO interface and provides a scatter-gather type capability for packet data storage.

The DMA implements packet buffering where dual-port memories are used to buffer multiple frames.

Downloaded from Arrow.com.

38.6.3.1 Packet Buffer DMA

- Easier to guarantee maximum line rate due to the ability to store multiple frames in the packet buffer, where the number of frames is limited by the amount of packet buffer memory and Ethernet frame size
- Full store and forward, or partial store and forward programmable options (partial store will cater for shorter latency requirements)
- Support for Transmit TCP/IP checksum offload
- Support for priority queueing
- When a collision on the line occurs during transmission, the packet will be automatically replayed directly from the packet buffer memory rather than having to re-fetch through the AHB (full store and forward ONLY)
- Received errored packets are automatically dropped before any of the packet is presented to the AHB (full store and forward ONLY), thus reducing AHB activity
- Supports manual RX packet flush capabilities
- Optional RX packet flush when there is lack of AHB resource

38.6.3.2 Partial Store and Forward Using Packet Buffer DMA

The DMA uses SRAM-based packet buffers, and can be programmed into a low latency mode, known as Partial Store and Forward. This allows for a reduced latency as the full packet is not buffered before forwarding. Note that this option is only available when the device is configured for full duplex operation. This feature is enabled via the programmable TX and RX Partial Store and Forward registers (GMAC_TPSF and GMAC_RPSF). When the transmit Partial Store and Forward mode is activated, the transmitter will only begin to forward the packet to the MAC when there is enough packet data stored in the packet buffer. Likewise, when the receive Partial Store and Forward mode is activated, the receiver will only begin to forward the packet to the AHB when enough packet data is stored in the packet buffer. The amount of packet data required to activate the forwarding process is programmable via watermark registers which are located at the same address as the partial store and forward enable bits. Note that the minimum operational value for the TX partial store and forward watermark is 20. There is no operational limit for the RX partial store and forward watermark. Enabling partial store and forward is a useful means to reduce latency, but there are performance implications. The GMAC DMA uses separate transmit and receive lists of buffer descriptors, with each descriptor describing a buffer area in memory. This allows Ethernet packets to be broken up and scattered around the AHB memory space.

38.6.3.3 Receive AHB Buffers

Received frames, optionally including FCS, are written to receive AHB buffers stored in memory. The receive buffer depth is programmable in the range of 64 bytes to 16 Kbytes through the DMA Configuration register (GMAC_DCFGR), with the default being 128 bytes.

The start location for each receive AHB buffer is stored in memory in a list of receive buffer descriptors at an address location pointed to by the receive buffer queue pointer. The base address for the receive buffer queue pointer is configured in software using the Receive Buffer Queue Base Address register (GMAC_RBQB).

Each list entry consists of two words. The first is the address of the receive AHB buffer and the second the receive status. If the length of a receive frame exceeds the AHB buffer length, the status word for the used buffer is written with zeroes except for the “start of frame” bit, which is always set for the first buffer in a frame. Bit zero of the address field is written to 1 to show the buffer has been used. The receive buffer manager then reads the location of the next receive AHB buffer and fills that with the next part of the received frame data. AHB buffers are filled until the frame is complete and the final buffer descriptor status word contains the complete frame status. Refer to Table 38-4 for details of the receive buffer descriptor list.

Each receive AHB buffer start location is a word address. The start of the first AHB buffer in a frame can be offset by up to three bytes, depending on the value written to bits 14 and 15 of the Network Configuration register (GMAC_NCFGR). If the start location of the AHB buffer is offset, the available length of the first AHB buffer is reduced by the corresponding number of bytes.

Downloaded from Arrow.com.

Table 38-4. Receive Buffer Descriptor Entry

Bit	Function
	Word 0
31:2	Address of beginning of buffer
1	Wrap—marks last descriptor in receive buffer descriptor list.
0	Ownership—needs to be zero for the GMAC to write data to the receive buffer. The GMAC sets this to one once it has successfully written a frame to memory. Software has to clear this bit before the buffer can be used again.
	Word 1
31	Global all ones broadcast address detected
30	Multicast hash match
29	Unicast hash match
28	–
27	Specific Address Register match found, bit 25 and bit 26 indicate which Specific Address Register causes the match.
26:25	Specific Address Register match. Encoded as follows: 00: Specific Address Register 1 match 01: Specific Address Register 2 match 10: Specific Address Register 3 match 11: Specific Address Register 4 match If more than one specific address is matched only one is indicated with priority 4 down to 1.
24	This bit has a different meaning depending on whether RX checksum offloading is enabled. **With RX checksum offloading disabled:** (bit 24 clear in Network Configuration Register) Type ID register match found, bit 22 and bit 23 indicate which type ID register causes the match. **With RX checksum offloading enabled:** (bit 24 set in Network Configuration Register) 0: The frame was not SNAP encoded and/or had a VLAN tag with the Canonical Format Indicator (CFI) bit set. 1: The frame was SNAP encoded and had either no VLAN tag or a VLAN tag with the CFI bit not set.
23:22	This bit has a different meaning depending on whether RX checksum offloading is enabled. **With RX checksum offloading disabled:** (bit 24 clear in Network Configuration) Type ID register match. Encoded as follows: 00: Type ID register 1 match 01: Type ID register 2 match 10: Type ID register 3 match 11: Type ID register 4 match If more than one Type ID is matched only one is indicated with priority 4 down to 1. **With RX checksum offloading enabled:** (bit 24 set in Network Configuration Register) 00: Neither the IP header checksum nor the TCP/UDP checksum was checked. 01: The IP header checksum was checked and was correct. Neither the TCP nor UDP checksum was checked. 10: Both the IP header and TCP checksum were checked and were correct. 11: Both the IP header and UDP checksum were checked and were correct.
21	VLAN tag detected—type ID of 0x8100. For packets incorporating the stacked VLAN processing feature, this bit will be set if the second VLAN tag has a type ID of 0x8100

Downloaded from Arrow.com.

Table 38-4. Receive Buffer Descriptor Entry (Continued)

Bit	Function
20	Priority tag detected—type ID of 0x8100 and null VLAN identifier. For packets incorporating the stacked VLAN processing feature, this bit will be set if the second VLAN tag has a type ID of 0x8100 and a null VLAN identifier.
19:17	VLAN priority—only valid if bit 21 is set.
16	Canonical format indicator (CFI) bit (only valid if bit 21 is set).
15	End of frame—when set the buffer contains the end of a frame. If end of frame is not set, then the only valid status bit is start of frame (bit 14).
14	Start of frame—when set the buffer contains the start of a frame. If both bits 15 and 14 are set, the buffer contains a whole frame.
13	This bit has a different meaning depending on whether jumbo frames and ignore FCS modes are enabled. If neither mode is enabled this bit will be zero. **With jumbo frame mode enabled:** (bit 3 set in Network Configuration Register) Additional bit for length of frame (bit[13]), that is concatenated with bits[12:0] **With ignore FCS mode enabled and jumbo frames disabled:** (bit 26 set in Network Configuration Register and bit 3 clear in Network Configuration Register) This indicates per frame FCS status as follows: 0: Frame had good FCS 1: Frame had bad FCS, but was copied to memory as ignore FCS enabled.
12:0	These bits represent the length of the received frame which may or may not include FCS depending on whether FCS discard mode is enabled. **With FCS discard mode disabled:** (bit 17 clear in Network Configuration Register) Least significant 12 bits for length of frame including FCS. If jumbo frames are enabled, these 12 bits are concatenated with bit[13] of the descriptor above. **With FCS discard mode enabled:** (bit 17 set in Network Configuration Register) Least significant 12 bits for length of frame excluding FCS. If jumbo frames are enabled, these 12 bits are concatenated with bit[13] of the descriptor above.

To receive frames, the AHB buffer descriptors must be initialized by writing an appropriate address to bits 31:2 in the first word of each list entry. Bit 0 must be written with zero. Bit 1 is the wrap bit and indicates the last entry in the buffer descriptor list.

The start location of the receive buffer descriptor list must be written with the receive buffer queue base address before reception is enabled (receive enable in the Network Control (GMAC_NCR) register). Once reception is enabled, any writes to the Receive Buffer Queue Base Address register are ignored. When read, it will return the current pointer position in the descriptor list, though this is only valid and stable when receive is disabled.

If the filter block indicates that a frame should be copied to memory, the receive data DMA operation starts writing data into the receive buffer. If an error occurs, the buffer is recovered.

An internal counter within the GMAC represents the receive buffer queue pointer and it is not visible through the CPU interface. The receive buffer queue pointer increments by two words after each buffer has been used. It re-initializes to the receive buffer queue base address if any descriptor has its wrap bit set.

As receive AHB buffers are used, the receive AHB buffer manager sets bit zero of the first word of the descriptor to logic one indicating the AHB buffer has been used.

Software should search through the “used” bits in the AHB buffer descriptors to find out how many frames have been received, checking the start of frame and end of frame bits.

When the DMA is configured in the packet buffer Partial Store And Forward mode, received frames are written out to the AHB buffers as soon as enough frame data exists in the packet buffer. For both cases, this may mean

Downloaded from Arrow.com.

several full AHB buffers are used before some error conditions can be detected. If a receive error is detected the receive buffer currently being written will be recovered. Previous buffers will not be recovered. As an example, when receiving frames with cyclic redundancy check (CRC) errors or excessive length, it is possible that a frame fragment might be stored in a sequence of AHB receive buffers. Software can detect this by looking for start of frame bit set in a buffer following a buffer with no end of frame bit set.

To function properly, a 10/100 Ethernet system should have no excessive length frames or frames greater than 128 bytes with CRC errors. Collision fragments will be less than 128 bytes long, therefore it will be a rare occurrence to find a frame fragment in a receive AHB buffer, when using the default value of 128 bytes for the receive buffers size.

When in packet buffer full store and forward mode, only good received frames are written out of the DMA, so no fragments will exist in the AHB buffers due to MAC receiver errors. There is still the possibility of fragments due to DMA errors, for example used bit read on the second buffer of a multi-buffer frame.

If bit zero of the receive buffer descriptor is already set when the receive buffer manager reads the location of the receive AHB buffer, then the buffer has been already used and cannot be used again until software has processed the frame and cleared bit zero. In this case, the “buffer not available” bit in the receive status register is set and an interrupt triggered. The receive resource error statistics register is also incremented.

When the DMA is configured in the packet buffer full store and forward mode, the user can optionally select whether received frames should be automatically discarded when no AHB buffer resource is available. This feature is selected via bit 24 of the DMA Configuration register (by default, the received frames are not automatically discarded). If this feature is off, then received packets will remain to be stored in the SRAM-based packet buffer until AHB buffer resource next becomes available. This may lead to an eventual packet buffer overflow if packets continue to be received when bit zero (used bit) of the receive buffer descriptor remains set. Note that after a used bit has been read, the receive buffer manager will re-read the location of the receive buffer descriptor every time a new packet is received. When the DMA is not configured in the packet buffer full store and forward mode and a used bit is read, the frame currently being received will be automatically discarded.

When the DMA is configured in the packet buffer full store and forward mode, a receive overrun condition occurs when the receive SRAM-based packet buffer is full, or because HRESP was not OK. In all other modes, a receive overrun condition occurs when either the AHB bus was not granted quickly enough, or because HRESP was not OK, or because a new frame has been detected by the receive block, but the status update or write back for the previous frame has not yet finished. For a receive overrun condition, the receive overrun interrupt is asserted and the buffer currently being written is recovered. The next frame that is received whose address is recognized reuses the buffer.

In any packet buffer mode, a write to bit 18 of GMAC_NCR will force a packet from the external SRAM-based receive packet buffer to be flushed. This feature is only acted upon when the RX DMA is not currently writing packet data out to AHB, i.e., it is in an IDLE state. If the RX DMA is active, a write to this bit is ignored.

38.6.3.4 Transmit AHB Buffers

Frames to transmit are stored in one or more transmit AHB buffers. Transmit frames can be between 1 and 16384 bytes long, so it is possible to transmit frames longer than the maximum length specified in the IEEE 802.3 standard. It should be noted that zero length AHB buffers are allowed and that the maximum number of buffers permitted for each transmit frame is 128.

The start location for each transmit AHB buffer is stored in memory in a list of transmit buffer descriptors at a location pointed to by the transmit buffer queue pointer. The base address for this queue pointer is set in software using the Transmit Buffer Queue Base Address register. Each list entry consists of two words. The first is the byte address of the transmit buffer and the second containing the transmit control and status. For the packet buffer DMA, the start location for each AHB buffer is a byte address, the bottom bits of the address being used to offset the start of the data from the data-word boundary (i.e., bits 2,1 and 0 are used to offset the address for 64-bit datapaths).

Downloaded from Arrow.com.

Frames can be transmitted with or without automatic CRC generation. If CRC is automatically generated, pad will also be automatically generated to take frames to a minimum length of 64 bytes. When CRC is not automatically generated (as defined in word 1 of the transmit buffer descriptor), the frame is assumed to be at least 64 bytes long and pad is not generated.

An entry in the transmit buffer descriptor list is described in Table 38-5.

To transmit frames, the buffer descriptors must be initialized by writing an appropriate byte address to bits [31:0] in the first word of each descriptor list entry.

The second word of the transmit buffer descriptor is initialized with control information that indicates the length of the frame, whether or not the MAC is to append CRC and whether the buffer is the last buffer in the frame.

After transmission the status bits are written back to the second word of the first buffer along with the used bit. Bit 31 is the used bit which must be zero when the control word is read if transmission is to take place. It is written to one once the frame has been transmitted. Bits[29:20] indicate various transmit error conditions. Bit 30 is the wrap bit which can be set for any buffer within a frame. If no wrap bit is encountered the queue pointer continues to increment.

The Transmit Buffer Queue Base Address register can only be updated while transmission is disabled or halted; otherwise any attempted write will be ignored. When transmission is halted the transmit buffer queue pointer will maintain its value. Therefore when transmission is restarted the next descriptor read from the queue will be from immediately after the last successfully transmitted frame. while transmit is disabled (bit 3 of the Network Control register set low), the transmit buffer queue pointer resets to point to the address indicated by the Transmit Buffer Queue Base Address register. Note that disabling receive does not have the same effect on the receive buffer queue pointer.

Once the transmit queue is initialized, transmit is activated by writing to the transmit start bit (bit 9) of the Network Control register. Transmit is halted when a buffer descriptor with its used bit set is read, a transmit error occurs, or by writing to the transmit halt bit of the Network Control register. Transmission is suspended if a pause frame is received while the pause enable bit is set in the Network Configuration register. Rewriting the start bit while transmission is active is allowed. This is implemented with TXGO variable which is readable in the Transmit Status register at bit location 3. The TXGO variable is reset when:

- Transmit is disabled.
- A buffer descriptor with its ownership bit set is read.
- Bit 10, THALT, of the Network Control register is written.
- There is a transmit error such as too many retries or a transmit underrun.

To set TXGO, write TSTART to the bit 9 of the Network Control register. Transmit halt does not take effect until any ongoing transmit finishes.

If the DMA is configured for packet buffer Partial Store and Forward mode and a collision occurs during transmission of a multi-buffer frame, transmission will automatically restart from the first buffer of the frame. For packet buffer mode, the entire contents of the frame are read into the transmit packet buffer memory, so the retry attempt will be replayed directly from the packet buffer memory rather than having to re-fetch through the AHB.

If a used bit is read midway through transmission of a multi-buffer frame, this is treated as a transmit error. Transmission stops, GTXER is asserted and the FCS will be bad.

If transmission stops due to a transmit error or a used bit being read, transmission restarts from the first buffer descriptor of the frame being transmitted when the transmit start bit is rewritten.

Downloaded from Arrow.com.

Table 38-5. Transmit Buffer Descriptor Entry

Bit	Function
Word 0	
31:0	Byte address of buffer
Word 1	
31	Used—must be zero for the GMAC to read data to the transmit buffer. The GMAC sets this to one for the first buffer of a frame once it has been successfully transmitted. Software must clear this bit before the buffer can be used again.
30	Wrap—marks last descriptor in transmit buffer descriptor list. This can be set for any buffer within the frame.
29	Retry limit exceeded, transmit error detected
28	Reserved.
27	Transmit frame corruption due to AHB error—set if an error occurs while midway through reading transmit frame from the AHB, including HRESP errors and buffers exhausted mid frame (if the buffers run out during transmission of a frame then transmission stops, FCS shall be bad and GTXER asserted). Also set if single frame is too large for configured packet buffer memory size.
26	Late collision, transmit error detected.
25:23	Reserved
22:20	Transmit IP/TCP/UDP checksum generation offload errors: 000: No Error. 001: The Packet was identified as a VLAN type, but the header was not fully complete, or had an error in it. 010: The Packet was identified as a SNAP type, but the header was not fully complete, or had an error in it. 011: The Packet was not of an IP type, or the IP packet was invalidly short, or the IP was not of type IPv4/IPv6. 100: The Packet was not identified as VLAN, SNAP or IP. 101: Non supported packet fragmentation occurred. For IPv4 packets, the IP checksum was generated and inserted. 110: Packet type detected was not TCP or UDP. TCP/UDP checksum was therefore not generated. For IPv4 packets, the IP checksum was generated and inserted. 111: A premature end of packet was detected and the TCP/UDP checksum could not be generated.
19:17	Reserved
16	No CRC to be appended by MAC. When set, this implies that the data in the buffers already contains a valid CRC, hence no CRC or padding is to be appended to the current frame by the MAC. This control bit must be set for the first buffer in a frame and will be ignored for the subsequent buffers of a frame. Note that this bit must be clear when using the transmit IP/TCP/UDP checksum generation offload, otherwise checksum generation and substitution will not occur.
15	Last buffer, when set this bit will indicate the last buffer in the current frame has been reached.
14	Reserved
13:0	Length of buffer

38.6.3.5 DMA Bursting on the AHB

The DMA will always use SINGLE, or INCR type AHB accesses for buffer management operations. When performing data transfers, the AHB burst length used can be programmed using bits 4:0 of the DMA Configuration register so that either SINGLE, INCR or fixed length incrementing bursts (INCR4, INCR8 or INCR16) are used where possible.

When there is enough space and enough data to be transferred, the programmed fixed length bursts will be used. If there is not enough data or space available, for example when at the beginning or the end of a buffer, SINGLE

Downloaded from Arrow.com.

type accesses are used. Also SINGLE type accesses are used at 1024 byte boundaries, so that the 1 Kbyte boundaries are not burst over as per AHB requirements.

The DMA will not terminate a fixed length burst early, unless an error condition occurs on the AHB or if receive or transmit are disabled in the Network Control register.

38.6.3.6 DMA Packet Buffer

The DMA uses packet buffers for both transmit and receive paths. This mode allows multiple packets to be buffered in both transmit and receive directions. This allows the DMA to withstand far greater access latencies on the AHB and make more efficient use of the AHB bandwidth. There are two modes of operation—Full Store and Forward and Partial Store and Forward.

As described above (Section 38.6.3.2 ”Partial Store and Forward Using Packet Buffer DMA”), the DMA can be programmed into a low latency mode, known as Partial Store and Forward. For further details of this mode, see Section 38.6.3.2.

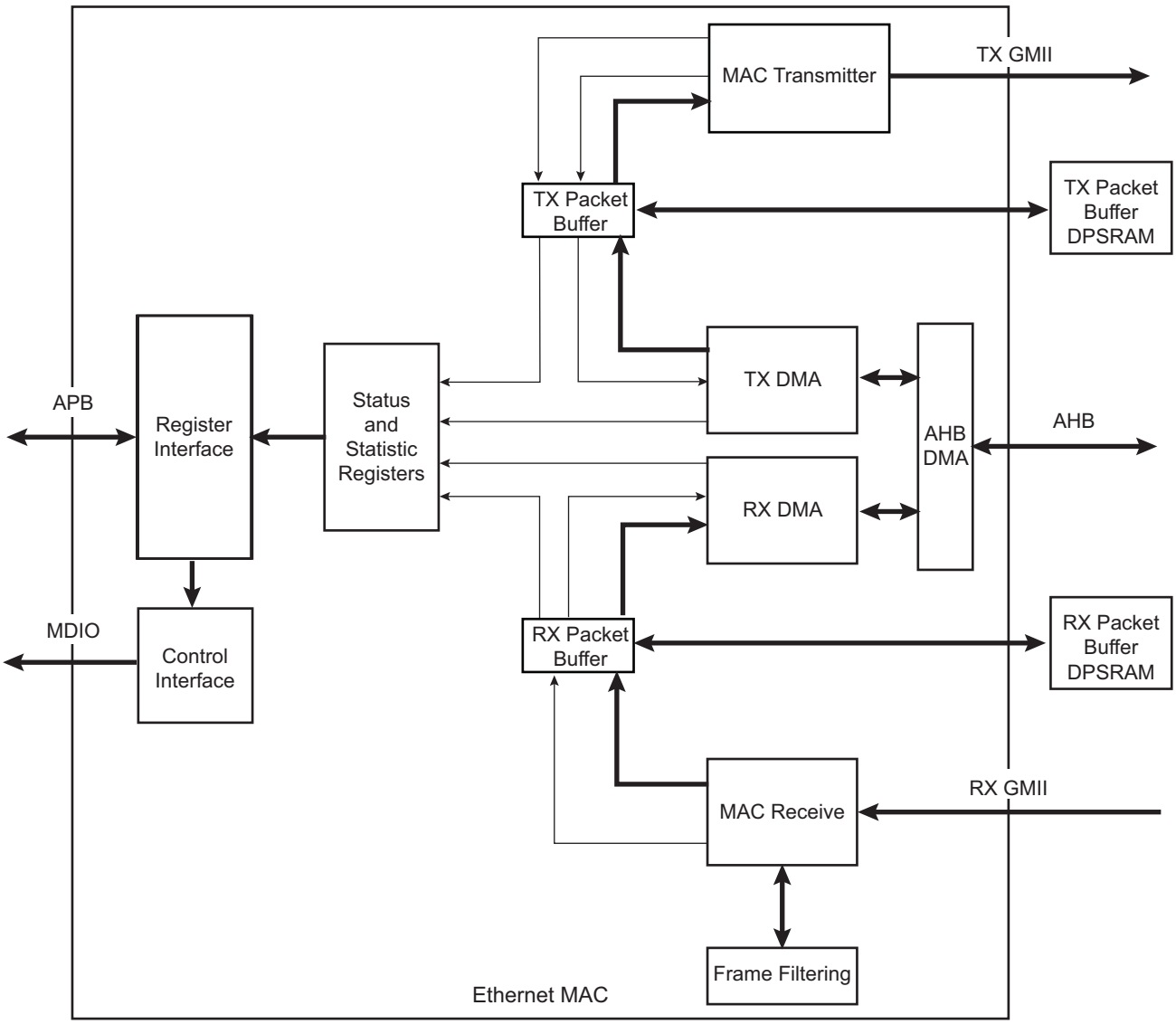
When the DMA is in full store and forward mode, full packets are buffered which provides the possibility to:

- Discard packets with error on the receive path before they are partially written out of the DMA, thus saving AHB bus bandwidth and driver processing overhead,
- Retry collided transmit frames from the buffer, thus saving AHB bus bandwidth,
- Implement transmit IP/TCP/UDP checksum generation offload.

With the packet buffers included, the structure of the GMAC data paths is shown in Figure 38-2.

Downloaded from Arrow.com.

Figure 38-2. Data Paths with Packet Buffers Included



38.6.3.7 Transmit Packet Buffer

The transmitter packet buffer will continue attempting to fetch frame data from the AHB system memory until the packet buffer itself is full, at which point it will attempt to maintain its full level.

To accommodate the status and statistics associated with each frame, three words per packet (or two if the GMAC is configured in 64-bit datapath mode) are reserved at the end of the packet data. If the packet is bad and requires to be dropped, the status and statistics are the only information held on that packet. Storing the status in the DPRAM is required in order to decouple the DMA interface of the buffer from the MAC interface, to update the MAC status/statistics and to generate interrupts in the order in which the packets that they represent were fetched from the AHB memory.

If any errors occur on the AHB while reading the transmit frame, the fetching of packet data from AHB memory is halted. The MAC transmitter will continue to fetch packet data, thereby emptying the packet buffer and allowing any good non-errored frames to be transmitted successfully. Once these have been fully transmitted, the status/statistics for the errored frame will be updated and software will be informed via an interrupt that an AHB error occurred. This way, the error is reported in the correct packet order.

Downloaded from Arrow.com.

The transmit packet buffer will only attempt to read more frame data from the AHB when space is available in the packet buffer memory. If space is not available it must wait until the a packet fetched by the MAC completes transmission and is subsequently removed from the packet buffer memory. Note that if full store and forward mode is active and if a single frame is fetched that is too large for the packet buffer memory, the frame is flushed and the DMA halted with an error status. This is because a complete frame must be written into the packet buffer before transmission can begin, and therefore the minimum packet buffer memory size should be chosen to satisfy the maximum frame to be transmitted in the application.

In full store and forward mode, once the complete transmit frame is written into the packet buffer memory, a trigger is sent across to the MAC transmitter, which will then begin reading the frame from the packet buffer memory. Since the whole frame is present and stable in the packet buffer memory an underflow of the transmitter is not possible. The frame is kept in the packet buffer until notification is received from the MAC that the frame data has either been successfully transmitted or can no longer be retransmitted (too many retries in half duplex mode). When this notification is received the frame is flushed from memory to make room for a new frame to be fetched from AHB system memory.

In Partial Store and Forward mode, a trigger is sent across to the MAC transmitter as soon as sufficient packet data is available, which will then begin fetching the frame from the packet buffer memory. If, after this point, the MAC transmitter is able to fetch data from the packet buffer faster than the AHB DMA can fill it, an underflow of the transmitter is possible. In this case, the transmission is terminated early, and the packet buffer is completely flushed. Transmission can only be restarted by writing to the transmit START bit.

In half duplex mode, the frame is kept in the packet buffer until notification is received from the MAC that the frame data has either been successfully transmitted or can no longer be retransmitted (too many retries in half duplex mode). When this notification is received the frame is flushed from memory to make room for a new frame to be fetched from AHB system memory.

In full duplex mode, the frame is removed from the packet buffer on the fly.

Other than underflow, the only MAC related errors that can occur are due to collisions during half duplex transmissions. When a collision occurs the frame still exists in the packet buffer memory so can be retried directly from there. Only once the MAC transmitter has failed to transmit after sixteen attempts is the frame finally flushed from the packet buffer.

38.6.3.8 Receive Packet Buffer

The receive packet buffer stores frames from the MAC receiver along with their status and statistics. Frames with errors are flushed from the packet buffer memory, while good frames are pushed onto the DMA AHB interface.

The receiver packet buffer monitors the FIFO write interface from the MAC receiver and translates the FIFO pushes into packet buffer writes. At the end of the received frame the status and statistics are buffered so that the information can be used when the frame is read out. When programmed in full store and forward mode, if the frame has an error the frame data is immediately flushed from the packet buffer memory allowing subsequent frames to utilise the freed up space. The status and statistics for bad frames are still used to update the GMAC registers.

To accommodate the status and statistics associated with each frame, three words per packet (or two if configured in 64-bit datapath mode) are reserved at the end of the packet data. If the packet is bad and requires to be dropped, the status and statistics are the only information held on that packet.

The receiver packet buffer will also detect a full condition so that an overflow condition can be detected. If this occurs, subsequent packets are dropped and an RX overflow interrupt is raised.

For full store and forward, the DMA only begins packet fetches once the status and statistics for a frame are available. If the frame has a bad status due to a frame error, the status and statistics are passed on to the GMAC registers. If the frame has a good status, the information is used to read the frame from the packet buffer memory and burst onto the AHB using the DMA buffer management protocol. Once the last frame data has been transferred to the packet buffer, the status and statistics are updated to the GMAC registers.

Downloaded from Arrow.com.

If Partial Store and Forward mode is active, the DMA will begin fetching the packet data before the status is available. As soon as the status becomes available, the DMA will fetch this information as soon as possible before continuing to fetch the remainder of the frame. Once the last frame data has been transferred to the packet buffer, the status and statistics are updated to the GMAC registers.

38.6.3.9 Priority Queueing in the DMA

The DMA by default uses a single transmit and receive queue. This means the list of transmit/receive buffer descriptors point to data buffers associated with a single transmit/receive data stream. The GMAC can select up to 3 priority queues. Each queue has an independent list of buffer descriptors pointing to separate data streams.

In the transmit direction, higher priority queues are always serviced before lower priority queues, with Q0 as lowest priority and Q2 as highest priority. This strict priority scheme requires the user to ensure that high priority traffic is constrained so that lower priority traffic will have required bandwidth. The GMAC DMA will determine the next queue to service by initiating a sequence of buffer descriptor reads interrogating the ownership bits of each. The buffer descriptor corresponding to the highest priority queue is read first. As an example, if the ownership bit of this descriptor is set, then the DMA will progress to reading the 2nd highest priority queue's descriptor. If that ownership bit read of this lower priority queue is set, then the DMA will read the 3rd highest priority queue's descriptor. If all the descriptors return an ownership bit set, then a resource error has occurred, an interrupt is generated and transmission is automatically halted. Transmission can only be restarted by setting the START bit in the Network Control register. The GMAC DMA will need to identify the highest available queue to transmit from when the START bit in the Network Control register is written to and the TX is in a halted state, or when the last word of any packet has been fetched from external AHB memory.

The GMAC transmit DMA maximizes the effectiveness of priority queuing by ensuring that high priority traffic be transmitted as early as possible after being fetched from AHB. High priority traffic fetched from AHB will be pushed to the MAC layer, depending on traffic shaping being enabled and the associated credit value for that queue, before any lower priority traffic that may pre-exist in the transmit SRAM-based packet buffer. This is achieved by separating the transmit SRAM-based packet buffer into regions, one region per queue. The size of each region determines the amount of SRAM space allocated per queue.

For each queue, there is an associated Transmit Buffer Queue Base Address register. For the lowest priority queue (or the only queue when only one queue is selected), the Transmit Buffer Queue Base Address is located at address 0x1C. For all other queues, the Transmit Buffer Queue Base Address registers are located at sequential addresses starting at address 0x440.

In the receive direction each packet is written to AHB data buffers in the order that it is received. For each queue, there is an independent set of receive AHB buffers for each queue. There is therefore a separate Receive Buffer Queue Base Address register for each queue. For the lowest priority queue (or the only queue when only one queue is selected), the Receive Buffer Queue Base Address is located at address 0x18. For all other queues, the Receive Buffer Queue Base Address registers are located at sequential addresses starting at address 0x480. Every received packet will pass through a programmable screening algorithm which will allocate a particular queue to that frame. The user interface to the screeners is through two types of programmable registers:

- Screening Type 1 registers—The module features 4 Screening Type 1 registers. Screening Type 1 registers hold values to match against specific IP and UDP fields of the received frames. The fields matched against are DS (Differentiated Services field of IPv4 frames), TC (Traffic class field of IPv6 frames) and/or the UDP destination port.
- Screening Type 2 registers—The module features 8 Screening Type 2 registers GMAC_ST2RPQ. Screening Type 2 registers operate independently of Screening Type 1 registers and offer additional match capabilities. Screening Type 2 allows a screen to be configured that is the combination of all or any of the following comparisons:

Downloaded from Arrow.com.

1. An enable bit VLAN priority, VLANE. A VLAN priority match will be performed if the VLAN priority enable is set. The extracted priority field in the VLAN header is compared against VLANP in the GMAC_ST2RPQ register itself.
2. An enable bit EtherType, ETHE. The EtherType field I2ETH inside GMAC_ST2RPQ maps to one of 4 EtherType match registers, GMAC_ST2ER. The extracted EtherType is compared against GMAC_ST2ER designated by this EtherType field.
3. An enable bit Compare A, COMPAE. This bit is associated with a Screening Type 2 Compare Word 0/1 register x, GMAC_ST2CW0/1.
4. An enable bit Compare B, COMPBE. This bit is associated with a Screening Type 2 Compare Word 0/1 register x, GMAC_ST2CW0/1.
5. An enable bit Compare C, COMPCE. This bit is associated with a Screening Type 2 Compare Word 0/1 register x, GMAC_ST2CW0/1.

Each screener type has an enable bit, a match pattern and a queue number. If a received frame matches on an enabled screening register, then the frame will be tagged with the queue value in the associated screening register, and forwarded onto the DMA and subsequently into the external memory associated with that queue. If two screeners are matched then the one which resides at the lowest register address will take priority so care must be taken on the selection of the screener location.

When the priority queuing feature is enabled, the number of interrupt outputs from the GMAC core is increased to match the number of supported queues. The number of Interrupt Status registers is increased by the same number. Only DMA related events are reported using the individual interrupt outputs, as the GMAC can relate these events to specific queues. All other events generated within the GMAC are reported in the interrupt associated with the lowest priority queue. For the lowest priority queue (or the only queue when only 1 queue is selected), the Interrupt Status register is located at address 0x24. For all other queues, the Interrupt Status register is located at sequential addresses starting at address 0x400.

Note: The address matching is the first level of filtering. If there is a match, the screeners are the next level of filtering for routing the data to the appropriate queue. See Section 38.6.7 ”MAC Filtering Block” for more details.

The additional screening done by the functions Compare A, B, and C each have an enable bit and compare register field. COMPA, COMPB and COMPC in GMAC_ST2RPQ are pointers to a configured offset (OFFSVAL), value (COMPVAL), and mask (MASKVAL). If enabled, the compare is true if the data at the offset into the frame, ANDed with MASKVAL, is equal to the value of COMPVAL ANDed with MASKVAL. A 16-bit word comparison is done. The byte at the offset number of bytes from the index start is compared to bits 7:0 of the configured COMPVAL and MASKVAL. The byte at the offset number of bytes + 1 from the index start is compared to bits 15:8 of the configured COMPVAL and MASKVAL.

The offset value in bytes, OFFSVAL, ranges from 0 to 127 bytes from either the start of the frame, the byte after the EtherType field, the byte after the IP header (IPv4 or IPv6) or the byte after the TCP/UDP header. Note the logic to decode the IP header or the TCP/UDP header is reused from the TCP/UDP/IP checksum offload logic and therefore has the same restrictions on use (the main limitation is that IP fragmentation is not supported). Refer to the Checksum Offload for IP, TCP and UDP section of this documentation for further details.

Compare A, B, and C use a common set of 24 GMAC_ST2CW0/1 registers, thus all COMPA, COMPB and COMPC fields in the registers GMAC_ST2RPQ point to a single pool of 24 GMAC_ST2CW0/1 registers.

Note that Compare A, B and C together allow matching against an arbitrary 48 bits of data and so can be used to match against a MAC address.

All enabled comparisons are ANDed together to form the overall type 2 screening match.

38.6.4 MAC Transmit Block

The MAC transmitter can operate in either half duplex or full duplex mode and transmits frames in accordance with the Ethernet IEEE 802.3 standard. In half duplex mode, the CSMA/CD protocol of the IEEE 802.3 specification is followed.

Downloaded from Arrow.com.

A small input buffer receives data through the FIFO interface which will extract data in 32-bit form. All subsequent processing prior to the final output is performed in bytes.

Transmit data can be output using the MII interface.

Frame assembly starts by adding preamble and the start frame delimiter. Data is taken from the transmit FIFO interface a word at a time.

If necessary, padding is added to take the frame length to 60 bytes. CRC is calculated using an order 32-bit polynomial. This is inverted and appended to the end of the frame taking the frame length to a minimum of 64 bytes. If the no CRC bit is set in the second word of the last buffer descriptor of a transmit frame, neither pad nor CRC are appended. The no CRC bit can also be set through the FIFO interface.

In full duplex mode (at all data rates), frames are transmitted immediately. Back to back frames are transmitted at least 96 bit times apart to guarantee the interframe gap.

In half duplex mode, the transmitter checks carrier sense. If asserted, the transmitter waits for the signal to become inactive, and then starts transmission after the interframe gap of 96 bit times. If the collision signal is asserted during transmission, the transmitter will transmit a jam sequence of 32 bits taken from the data register and then retry transmission after the back off time has elapsed. If the collision occurs during either the preamble or Start Frame Delimiter (SFD), then these fields will be completed prior to generation of the jam sequence.

The back off time is based on an XOR of the 10 least significant bits of the data coming from the transmit FIFO interface and a 10-bit pseudo random number generator. The number of bits used depends on the number of collisions seen. After the first collision 1 bit is used, then the second 2 bits and so on up to the maximum of 10 bits. All 10 bits are used above ten collisions. An error will be indicated and no further attempts will be made if 16 consecutive attempts cause collision. This operation is compliant with the description in Clause 4.2.3.2.5 of the IEEE 802.3 standard which refers to the truncated binary exponential back off algorithm.

In 10/100 mode, both collisions and late collisions are treated identically, and back off and retry will be performed up to 16 times. This condition is reported in the transmit buffer descriptor word 1 (late collision, bit 26) and also in the Transmit Status register (late collision, bit 7). An interrupt can also be generated (if enabled) when this exception occurs, and bit 5 in the Interrupt Status register will be set.

In all modes of operation, if the transmit DMA underruns, a bad CRC is automatically appended using the same mechanism as jam insertion and the GTXER signal is asserted. For a properly configured system this should never happen and also it is impossible if configured to use the DMA with packet buffers, as the complete frame is buffered in local packet buffer memory.

By setting when bit 28 is set in the Network Configuration register, the Inter Packet Gap (IPG) may be stretched beyond 96 bits depending on the length of the previously transmitted frame and the value written to the IPG Stretch register (GMAC_IPGS). The least significant 8 bits of the IPG Stretch register multiply the previous frame length (including preamble). The next significant 8 bits (+1 so as not to get a divide by zero) divide the frame length to generate the IPG. IPG stretch only works in full duplex mode and when bit 28 is set in the Network Configuration register. The IPG Stretch register cannot be used to shrink the IPG below 96 bits.

If the back pressure bit is set in the Network Control register, or if the HDFC configuration bit is set in the GMAC_UR register (10M or 100M half duplex mode), the transmit block transmits 64 bits of data, which can consist of 16 nibbles of 1011 or in bit rate mode 64 1s, whenever it sees an incoming frame to force a collision. This provides a way of implementing flow control in half duplex mode.

38.6.5 MAC Receive Block

All processing within the MAC receive block is implemented using a 16-bit data path. The MAC receive block checks for valid preamble, FCS, alignment and length, presents received frames to the FIFO interface and stores the frame destination address for use by the address checking block.

If, during the frame reception, the frame is found to be too long, a bad frame indication is sent to the FIFO interface. The receiver logic ceases to send data to memory as soon as this condition occurs.

Downloaded from Arrow.com.

At end of frame reception the receive block indicates to the DMA block whether the frame is good or bad. The DMA block will recover the current receive buffer if the frame was bad.

Ethernet frames are normally stored in DMA memory complete with the FCS. Setting the FCS remove bit in the network configuration (bit 17) causes frames to be stored without their corresponding FCS. The reported frame length field is reduced by four bytes to reflect this operation.

The receive block signals to the register block to increment the alignment, CRC (FCS), short frame, long frame, jabber or receive symbol errors when any of these exception conditions occur.

If bit 26 is set in the network configuration, CRC errors will be ignored and CRC errored frames will not be discarded, though the Frame Check Sequence Errors statistic register will still be incremented. Additionally, if not enabled for jumbo frames mode, then bit[13] of the receiver descriptor word 1 will be updated to indicate the FCS validity for the particular frame. This is useful for applications such as EtherCAT whereby individual frames with FCS errors must be identified.

Received frames can be checked for length field error by setting the length field error frame discard bit of the Network Configuration register (bit-16). When this bit is set, the receiver compares a frame's measured length with the length field (bytes 13 and 14) extracted from the frame. The frame is discarded if the measured length is shorter. This checking procedure is for received frames between 64 bytes and 1518 bytes in length.

Each discarded frame is counted in the 10-bit length field error statistics register. Frames where the length field is greater than or equal to 0x0600 hex will not be checked.

38.6.6 Checksum Offload for IP, TCP and UDP

The GMAC can be programmed to perform IP, TCP and UDP checksum offloading in both receive and transmit directions, which is enabled by setting bit 24 in the Network Configuration register for receive and bit 11 in the DMA Configuration register for transmit.

IPv4 packets contain a 16-bit checksum field, which is the 16-bit 1’s complement of the 1’s complement sum of all 16-bit words in the header. TCP and UDP packets contain a 16-bit checksum field, which is the 16-bit 1’s complement of the 1’s complement sum of all 16-bit words in the header, the data and a conceptual IP pseudo header.

To calculate these checksums in software requires each byte of the packet to be processed. For TCP and UDP this can use a large amount of processing power. Offloading the checksum calculation to hardware can result in significant performance improvements.

For IP, TCP or UDP checksum offload to be useful, the operating system containing the protocol stack must be aware that this offload is available so that it can make use of the fact that the hardware can either generate or verify the checksum.

38.6.6.1 Receiver Checksum Offload

When receive checksum offloading is enabled in the GMAC, the IPv4 header checksum is checked as per RFC 791, where the packet meets the following criteria:

- If present, the VLAN header must be four octets long and the CFI bit must not be set.
- Encapsulation must be RFC 894 Ethernet Type Encoding or RFC 1042 SNAP Encoding.
- IPv4 packet
- IP header is of a valid length

The GMAC also checks the TCP checksum as per RFC 793, or the UDP checksum as per RFC 768, if the following criteria are met:

- IPv4 or IPv6 packet
- Good IP header checksum (if IPv4)
- No IP fragmentation
- TCP or UDP packet

Downloaded from Arrow.com.

When an IP, TCP or UDP frame is received, the receive buffer descriptor gives an indication if the GMAC was able to verify the checksums. There is also an indication if the frame had SNAP encapsulation. These indication bits will replace the type ID match indication bits when the receive checksum offload is enabled. For details of these indication bits refer to Table 38-4 “Receive Buffer Descriptor Entry”.

If any of the checksums are verified as incorrect by the GMAC, the packet is discarded and the appropriate statistics counter incremented.

38.6.6.2 Transmitter Checksum Offload

The transmitter checksum offload is only available if the full store and forward mode is enabled. This is because the complete frame to be transmitted must be read into the packet buffer memory before the checksum can be calculated and written back into the headers at the beginning of the frame.

Transmitter checksum offload is enabled by setting bit [11] in the DMA Configuration register. When enabled, it will monitor the frame as it is written into the transmitter packet buffer memory to automatically detect the protocol of the frame. Protocol support is identical to the receiver checksum offload.

For transmit checksum generation and substitution to occur, the protocol of the frame must be recognized and the frame must be provided without the FCS field, by making sure that bit [16] of the transmit descriptor word 1 is clear. If the frame data already had the FCS field, this would be corrupted by the substitution of the new checksum fields.

If these conditions are met, the transmit checksum offload engine will calculate the IP, TCP and UDP checksums as appropriate. Once the full packet is completely written into packet buffer memory, the checksums will be valid and the relevant DPRAM locations will be updated for the new checksum fields as per standard IP/TCP and UDP packet structures.

If the transmitter checksum engine is prevented from generating the relevant checksums, bits [22:20] of the transmitter DMA writeback status will be updated to identify the reason for the error. Note that the frame will still be transmitted but without the checksum substitution, as typically the reason that the substitution did not occur was that the protocol was not recognized.

38.6.7 MAC Filtering Block

The filter block determines which frames should be written to the FIFO interface and on to the DMA.

Whether a frame is passed depends on what is enabled in the Network Configuration register, the state of the external matching pins, the contents of the specific address, type and Hash registers and the frame's destination address and type field.

If bit 25 of the Network Configuration register is not set, a frame will not be copied to memory if the GMAC is transmitting in half duplex mode at the time a destination address is received.

Ethernet frames are transmitted a byte at a time, least significant bit first. The first six bytes (48 bits) of an Ethernet frame make up the destination address. The first bit of the destination address, which is the LSB of the first byte of the frame, is the group or individual bit. This is one for multicast addresses and zero for unicast. The all ones address is the broadcast address and a special case of multicast.

The GMAC supports recognition of four specific addresses. Each specific address requires two registers, Specific Address register Bottom and Specific Address register Top. Specific Address register Bottom stores the first four bytes of the destination address and Specific Address register Top contains the last two bytes. The addresses stored can be specific, group, local or universal.

The destination address of received frames is compared against the data stored in the Specific Address registers once they have been activated. The addresses are deactivated at reset or when their corresponding Specific Address register Bottom is written. They are activated when Specific Address register Top is written. If a receive frame address matches an active address, the frame is written to the FIFO interface and on to DMA memory.

Downloaded from Arrow.com.

Frames may be filtered using the type ID field for matching. Four type ID registers exist in the register address space and each can be enabled for matching by writing a one to the MSB (bit 31) of the respective register. When a frame is received, the matching is implemented as an OR function of the various types of match.

The contents of each type ID register (when enabled) are compared against the length/type ID of the frame being received (e.g., bytes 13 and 14 in non-VLAN and non-SNAP encapsulated frames) and copied to memory if a match is found. The encoded type ID match bits (Word 0, Bit 22 and Bit 23) in the receive buffer descriptor status are set indicating which type ID register generated the match, if the receive checksum offload is disabled.

The reset state of the type ID registers is zero, hence each is initially disabled.

The following example illustrates the use of the address and type ID match registers for a MAC address of 21:43:65:87:A9:CB:

Preamble	55
SFD	D5
DA (Octet 0 - LSB)	21
DA (Octet 1)	43
DA (Octet 2)	65
DA (Octet 3)	87
DA (Octet 4)	A9
DA (Octet 5 - MSB)	CB
SA (LSB)	00(1)
SA	00(1)
SA	00(1)
SA	00(1)
SA	00(1)
SA (MSB)	00(1)
Type ID (MSB)	43
Type ID (LSB)	21

Note: 1. Contains the address of the transmitting device

The sequence above shows the beginning of an Ethernet frame. Byte order of transmission is from top to bottom as shown. For a successful match to specific address 1, the following address matching registers must be set up:

Specific Address 1 Bottom register (GMAC_SAB1) (Address 0x088) 0x87654321

Specific Address 1 Top register (GMAC_SAT1) (Address 0x08C) 0x0000CBA9

For a successful match to the type ID, the following Type ID Match 1 register must be set up:

Type ID Match 1 register (GMAC_TIDM1) (Address 0x0A8) 0x80004321

38.6.8 Broadcast Address

Frames with the broadcast address of 0xFFFFFFFFFFFF are stored to memory only if the 'no broadcast' bit in the Network Configuration register is set to zero.

38.6.9 Hash Addressing

The hash address register is 64 bits long and takes up two locations in the memory map. The least significant bits are stored in Hash Register Bottom and the most significant bits in Hash Register Top.

Downloaded from Arrow.com.

The unicast hash enable and the multicast hash enable bits in the Network Configuration register enable the reception of hash matched frames. The destination address is reduced to a 6-bit index into the 64-bit Hash register using the following hash function: The hash function is an XOR of every sixth bit of the destination address.

```
hash_index[05] = da[05] ^ da[11] ^ da[17] ^ da[23] ^ da[29] ^ da[35] ^ da[41] ^
da[47]
hash_index[04] = da[04] ^ da[10] ^ da[16] ^ da[22] ^ da[28] ^ da[34] ^ da[40] ^
da[46]
hash_index[03] = da[03] ^ da[09] ^ da[15] ^ da[21] ^ da[27] ^ da[33] ^ da[39] ^
da[45]
hash_index[02] = da[02] ^ da[08] ^ da[14] ^ da[20] ^ da[26] ^ da[32] ^ da[38] ^
da[44]
hash_index[01] = da[01] ^ da[07] ^ da[13] ^ da[19] ^ da[25] ^ da[31] ^ da[37] ^
da[43]
hash_index[00] = da[00] ^ da[06] ^ da[12] ^ da[18] ^ da[24] ^ da[30] ^ da[36] ^
da[42]
```

`da[0]`

represents the least significant bit of the first byte received, that is, the multicast/unicast indicator, and `da[47]` represents the most significant bit of the last byte received.

If the hash index points to a bit that is set in the Hash register then the frame will be matched according to whether the frame is multicast or unicast.

A multicast match will be signalled if the multicast hash enable bit is set, `da[0]` is logic 1 and the hash index points to a bit set in the Hash register.

A unicast match will be signalled if the unicast hash enable bit is set, `da[0]` is logic 0 and the hash index points to a bit set in the Hash register.

To receive all multicast frames, the Hash register should be set with all ones and the multicast hash enable bit should be set in the Network Configuration register.

38.6.10 Copy all Frames (Promiscuous Mode)

If the Copy All Frames bit is set in the Network Configuration register then all frames (except those that are too long, too short, have FCS errors or have GRXER asserted during reception) will be copied to memory. Frames with FCS errors will be copied if bit 26 is set in the Network Configuration register.

38.6.11 Disable Copy of Pause Frames

Pause frames can be prevented from being written to memory by setting the disable copying of pause frames control bit 23 in the Network Configuration register. When set, pause frames are not copied to memory regardless of the Copy All Frames bit, whether a hash match is found, a type ID match is identified or if a destination address match is found.

38.6.12 VLAN Support

The following table describes an Ethernet encoded 802.1Q VLAN tag.

Table 38-6. 802.1Q VLAN Tag

TPID (Tag Protocol Identifier) 16 bits	TCI (Tag Control Information) 16 bits
0x8100	First 3 bits priority, then CFI bit, last 12 bits VID

The VLAN tag is inserted at the 13th byte of the frame adding an extra four bytes to the frame. To support these extra four bytes, the GMAC can accept frame lengths up to 1536 bytes by setting bit 8 in the Network Configuration register.

Downloaded from Arrow.com.

If the VID (VLAN identifier) is null (0x000) this indicates a priority-tagged frame.

The following bits in the receive buffer descriptor status word give information about VLAN tagged frames:-

- Bit 21 set if receive frame is VLAN tagged (i.e., type ID of 0x8100).
- Bit 20 set if receive frame is priority tagged (i.e., type ID of 0x8100 and null VID). (If bit 20 is set, bit 21 will be set also.)
- Bit 19, 18 and 17 set to priority if bit 21 is set.
- Bit 16 set to CFI if bit 21 is set.

The GMAC can be configured to reject all frames except VLAN tagged frames by setting the discard non-VLAN frames bit in the Network Configuration register.

38.6.13 Wake on LAN Support

The receive block supports Wake on LAN by detecting the following events on incoming receive frames:

- Magic packet
- Address Resolution Protocol (ARP) request to the device IP address
- Specific address 1 filter match
- Multicast hash filter match

These events can be individually enabled through bits [19:16] of the Wake on LAN register. Also, for Wake on LAN detection to occur, receive enable must be set in the Network Control register, however a receive buffer does not have to be available.

In case of an ARP request, specific address 1 or multicast filter events will occur even if the frame is errored. For magic packet events, the frame must be correctly formed and error free.

A magic packet event is detected if all of the following are true:

- Magic packet events are enabled through bit 16 of the Wake on LAN register
- The frame's destination address matches specific address 1
- The frame is correctly formed with no errors
- The frame contains at least 6 bytes of 0xFF for synchronization
- There are 16 repetitions of the contents of Specific Address 1 register immediately following the synchronization

An ARP request event is detected if all of the following are true:

- ARP request events are enabled through bit 17 of the Wake on LAN register
- Broadcasts are allowed by bit 5 in the Network Configuration register
- The frame has a broadcast destination address (bytes 1 to 6)
- The frame has a type ID field of 0x0806 (bytes 13 and 14)
- The frame has an ARP operation field of 0x0001 (bytes 21 and 22)
- The least significant 16 bits of the frame's ARP target protocol address (bytes 41 and 42) match the value programmed in bits[15:0] of the Wake on LAN register

The decoding of the ARP fields adjusts automatically if a VLAN tag is detected within the frame. The reserved value of 0x0000 for the Wake on LAN target address value will not cause an ARP request event, even if matched by the frame.

A specific address 1 filter match event will occur if all of the following are true:

- Specific address 1 events are enabled through bit 18 of the Wake on LAN register
- The frame's destination address matches the value programmed in the Specific Address 1 registers

A multicast filter match event will occur if all of the following are true:

- Multicast hash events are enabled through bit 19 of the Wake on LAN register

Downloaded from Arrow.com.

- Multicast hash filtering is enabled through bit 6 of the Network Configuration register
- The frame destination address matches against the multicast hash filter
- The frame destination address is not a broadcast

38.6.14 IEEE 1588 Support

IEEE 1588 is a standard for precision time synchronization in local area networks. It works with the exchange of special Precision Time Protocol (PTP) frames. The PTP messages can be transported over IEEE 802.3/Ethernet, over Internet Protocol Version 4 or over Internet Protocol Version 6 as described in the annex of IEEE P1588.D2.1.

The GMAC indicates the message time-stamp point (asserted on the start packet delimiter and de-asserted at end of frame) for all frames and the passage of PTP event frames (asserted when a PTP event frame is detected and de-asserted at end of frame).

IEEE 802.1AS is a subset of IEEE 1588. One difference is that IEEE 802.1AS uses the Ethernet multicast address 0180C200000E for sync frame recognition whereas IEEE 1588 does not. GMAC is designed to recognize sync frames with both IEEE 802.1AS and IEEE 1588 addresses and so can support both 1588 and 802.1AS frame recognition simultaneously.

Synchronization between master and slave clocks is a two stage process.

First, the offset between the master and slave clocks is corrected by the master sending a sync frame to the slave with a follow up frame containing the exact time the sync frame was sent. Hardware assist modules at the master and slave side detect exactly when the sync frame was sent by the master and received by the slave. The slave then corrects its clock to match the master clock.

Second, the transmission delay between the master and slave is corrected. The slave sends a delay request frame to the master which sends a delay response frame in reply. Hardware assist modules at the master and slave side detect exactly when the delay request frame was sent by the slave and received by the master. The slave will now have enough information to adjust its clock to account for delay. For example, if the slave was assuming zero delay, the actual delay will be half the difference between the transmit and receive time of the delay request frame (assuming equal transmit and receive times) because the slave clock will be lagging the master clock by the delay time already.

The time-stamp is taken when the message time-stamp point passes the clock time-stamp point. This can generate an interrupt if enabled (GMAC_IER). However, MAC Filtering configuration is needed to actually ‘copy’ the message to memory. For Ethernet, the message time-stamp point is the SFD and the clock time-stamp point is the MII interface. (The IEEE 1588 specification refers to sync and delay_req messages as event messages as these require time-stamping. These events are captured in the registers GMAC_EFTx and GMAC_EFRx, respectively. Follow up, delay response and management messages do not require time-stamping and are referred to as general messages.)

1588 version 2 defines two additional PTP event messages. These are the peer delay request (Pdelay_Req) and peer delay response (Pdelay_Resp) messages. These events are captured in the registers GMAC_PEFTx and GMAC_PEFRx, respectively. These messages are used to calculate the delay on a link. Nodes at both ends of a link send both types of frames (regardless of whether they contain a master or slave clock). The Pdelay_Resp message contains the time at which a Pdelay_Req was received and is itself an event message. The time at which a Pdelay_Resp message is received is returned in a Pdelay_Resp_Follow_Up message.

1588 version 2 introduces transparent clocks of which there are two kinds, peer-to-peer (P2P) and end-to-end (E2E). Transparent clocks measure the transit time of event messages through a bridge and amend a correction field within the message to allow for the transit time. P2P transparent clocks additionally correct for the delay in the receive path of the link using the information gathered from the peer delay frames. With P2P transparent clocks delay_req messages are not used to measure link delay. This simplifies the protocol and makes larger systems more stable.

The GMAC recognizes four different encapsulations for PTP event messages:

Downloaded from Arrow.com.

1. 1588 version 1 (UDP/IPv4 multicast)
2. 1588 version 2 (UDP/IPv4 multicast)
3. 1588 version 2 (UDP/IPv6 multicast)
4. 1588 version 2 (Ethernet multicast)

Table 38-7. Example of Sync Frame in 1588 Version 1 Format

Frame Segment	Value
Preamble/SFD	55555555555555D5
DA (Octets 0–5)	—
SA (Octets 6–11)	—
Type (Octets 12–13)	0800
IP stuff (Octets 14–22)	—
UDP (Octet 23)	11
IP stuff (Octets 24–29)	—
IP DA (Octets 30–32)	E00001
IP DA (Octet 33)	81 or 82 or 83 or 84
Source IP port (Octets 34–35)	—
Dest IP port (Octets 36–37)	013F
Other stuff (Octets 38–42)	—
Version PTP (Octet 43)	01
Other stuff (Octets 44–73)	—
Control (Octet 74)	00
Other stuff (Octets 75–168)	—

Table 38-8. Example of Delay Request Frame in 1588 Version 1 Format

Frame Segment	Value
Preamble/SFD	55555555555555D5
DA (Octets 0–5)	—
SA (Octets 6–11)	—
Type (Octets 12–13)	0800
IP stuff (Octets 14–22)	—
UDP (Octet 23)	11
IP stuff (Octets 24–29)	—
IP DA (Octets 30–32)	E00001
IP DA (Octet 33)	81 or 82 or 83 or 84
Source IP port (Octets 34–35)	—
Dest IP port (Octets 36–37)	013F
Other stuff (Octets 38–42)	—
Version PTP (Octet 43)	01

Downloaded from Arrow.com.

Table 38-8. Example of Delay Request Frame in 1588 Version 1 Format (Continued)

Frame Segment	Value
Other stuff (Octets 44–73)	—
Control (Octet 74)	01
Other stuff (Octets 75–168)	—

For 1588 version 1 messages, sync and delay request frames are indicated by the GMAC if the frame type field indicates TCP/IP, UDP protocol is indicated, the destination IP address is 224.0.1.129/130/131 or 132, the destination UDP port is 319 and the control field is correct.

The control field is 0x00 for sync frames and 0x01 for delay request frames.

For 1588 version 2 messages, the type of frame is determined by looking at the message type field in the first byte of the PTP frame. Whether a frame is version 1 or version 2 can be determined by looking at the version PTP field in the second byte of both version 1 and version 2 PTP frames.

In version 2 messages sync frames have a message type value of 0x0, delay_req have 0x1, Pdelay_Req have 0x2 and Pdelay_Resp have 0x3.

Table 38-9. Example of Sync Frame in 1588 Version 2 (UDP/IPv4) Format

Frame Segment	Value
Preamble/SFD	55555555555555D5
DA (Octets 0–5)	—
SA (Octets 6–11)	—
Type (Octets 12–13)	0800
IP stuff (Octets 14–22)	—
UDP (Octet 23)	11
IP stuff (Octets 24–29)	—
IP DA (Octets 30–33)	E0000181
Source IP port (Octets 34–35)	—
Dest IP port (Octets 36–37)	013F
Other stuff (Octets 38–41)	—
Message type (Octet 42)	00
Version PTP (Octet 43)	02

Table 38-10. Example of Pdelay_Req Frame in 1588 Version 2 (UDP/IPv4) Format

Frame Segment	Value
Preamble/SFD	55555555555555D5
DA (Octets 0–5)	—
SA (Octets 6–11)	—
Type (Octets 12–13)	0800
IP stuff (Octets 14–22)	—
UDP (Octet 23)	11
IP stuff (Octets 24–29)	—

Downloaded from Arrow.com.

Table 38-10. Example of Pdelay_Req Frame in 1588 Version 2 (UDP/IPv4) Format (Continued)

Frame Segment	Value
IP DA (Octets 30–33)	E000006B
Source IP port (Octets 34–35)	—
Dest IP port (Octets 36–37)	013F
Other stuff (Octets 38–41)	—
Message type (Octet 42)	02
Version PTP (Octet 43)	02

Table 38-11. Example of Sync Frame in 1588 Version 2 (UDP/IPv6) Format

Frame Segment	Value
Preamble/SFD	55555555555555D5
DA (Octets 0–5)	—
SA (Octets 6–11)	—
Type (Octets 12–13)	86dd
IP stuff (Octets 14–19)	—
UDP (Octet 20)	11
IP stuff (Octets 21–37)	—
IP DA (Octets 38–53)	FF0X00000000018
Source IP port (Octets 54–55)	—
Dest IP port (Octets 56–57)	013F
Other stuff (Octets 58–61)	—
Message type (Octet 62)	00
Other stuff (Octets 63–93)	—
Version PTP (Octet 94)	02

Table 38-12. Example of Pdelay_Resp Frame in 1588 Version 2 (UDP/IPv6) Format

Frame Segment	Value
Preamble/SFD	55555555555555D5
DA (Octets 0–5)	—
SA (Octets 6–11)	—
Type (Octets 12–13)	86dd
IP stuff (Octets 14–19)	—
UDP (Octet 20)	11
IP stuff (Octets 21–37)	—
IP DA (Octets 38–53)	FF0200000000006B
Source IP port (Octets 54–55)	—
Dest IP port (Octets 56–57)	013F

Downloaded from Arrow.com.

Table 38-12. Example of Pdelay_Resp Frame in 1588 Version 2 (UDP/IPv6) Format (Continued)

Frame Segment	Value
Other stuff (Octets 58–61)	—
Message type (Octet 62)	03
Other stuff (Octets 63–93)	—
Version PTP (Octet 94)	02

For the multicast address 011B19000000 sync and delay request frames are recognized depending on the message type field, 00 for sync and 01 for delay request.

Table 38-13. Example of Sync Frame in 1588 Version 2 (Ethernet Multicast) Format

Frame Segment	Value
Preamble/SFD	55555555555555D5
DA (Octets 0–5)	011B19000000
SA (Octets 6–11)	—
Type (Octets 12–13)	88F7
Message type (Octet 14)	00
Version PTP (Octet 15)	02

Pdelay request frames need a special multicast address so they can pass through ports blocked by the spanning tree protocol. For the multicast address 0180C200000E sync, Pdelay_Req and Pdelay_Resp frames are recognized depending on the message type field, 00 for sync, 02 for pdelay request and 03 for pdelay response.

Table 38-14. Example of Pdelay_Req Frame in 1588 Version 2 (Ethernet Multicast) Format

Frame Segment	Value
Preamble/SFD	55555555555555D5
DA (Octets 0–5)	0180C200000E
SA (Octets 6–11)	—
Type (Octets 12–13)	88F7
Message type (Octet 14)	00
Version PTP (Octet 15)	02

38.6.15 Time Stamp Unit

The TSU consists of a timer and registers to capture the time at which PTP event frames cross the message timestamp point. An interrupt is issued when a capture register is updated.

The timer is implemented as a 94-bit register with the upper 48 bits counting seconds, the next 30 bits counting nanoseconds and the lowest 16 bits counting sub-nanoseconds. The lower 46 bits rolls over when they have counted to one second. An interrupt is generated when the seconds increment. The timer value can be read, written and adjusted through the APB interface. The timer is clocked by MCK.

The amount by which the timer increments each clock cycle is controlled by the timer increment registers (GMAC_TI). Bits 7:0 are the default increment value in nanoseconds and an additional 16 bits of sub-nanosecond resolution are available using the Timer Increment Sub-nanoseconds register (GMAC_TISUBN). If the rest of the register is written with zero, the timer increments by the value in [7:0], plus the value of GMAC_TISUBN, each clock cycle.

Downloaded from Arrow.com.

The GMAC_TISUBN register allows a resolution of approximately 15 femtoseconds.

Bits 15:8 of the increment register are the alternative increment value in nanoseconds and bits 23:16 are the number of increments after which the alternative increment value is used. If 23:16 are zero then the alternative increment value will never be used.

Taking the example of 10.2 MHz, there are 102 cycles every ten microseconds or 51 every five microseconds. So a timer with a 10.2 MHz clock source is constructed by incrementing by 98 ns for fifty cycles and then incrementing by 100 ns ($98 \times 50 + 100 = 5000$). This is programmed by setting the 1588 Timer Increment register to 0x00326462.

For a 49.8 MHz clock source it would be 20 ns for 248 cycles followed by an increment of 40 ns ($20 \times 248 + 40 = 5000$) programmed as 0x00F82814.

Having eight bits for the “number of increments” field allows frequencies up to 50 MHz to be supported with 200 kHz resolution.

Without the alternative increment field the period of the clock would be limited to an integer number of nanoseconds, resulting in supported clock frequencies of 8, 10, 20, 25, 40, 50, 100, 125, 200 and 250 MHz.

There are eight additional 80-bit registers that capture the time at which PTP event frames are transmitted and received. An interrupt is issued when these registers are updated. The TSU timer count value can be compared to a programmable comparison value. For the comparison, the 48 bits of the seconds value and the upper 22 bits of the nanoseconds value are used. A signal (GTSUCOMP) is provided to indicate when the TSU timer count value is equal to the comparison value stored in the TSU timer comparison value registers (0x0DC, 0x0E0, and 0x0E4). The GTSUCOMP signal can be routed to the Timer peripheral to automatically toggle pin TIOB11/PD22. This can be used as the reference clock for an external PLL to regenerate the audio clock in Ethernet AVB. An interrupt can also be generated (if enabled) when the TSU timer count value and comparison value are equal, mapped to bit 29 of the interrupt status register.

38.6.16 MAC 802.3 Pause Frame Support

Note: See Clause 31, and Annex 31A and 31B of the IEEE standard 802.3 for a full description of MAC 802.3 pause operation.

The following table shows the start of a MAC 802.3 pause frame.

Table 38-15. Start of an 802.3 Pause Frame

Address		Type (MAC Control Frame)	Pause	
Destination	Source		Opcode	Time
0x0180C2000001	6 bytes	0x8808	0x0001	2 bytes

The GMAC supports both hardware controlled pause of the transmitter, upon reception of a pause frame, and hardware generated pause frame transmission.

38.6.16.1 802.3 Pause Frame Reception

Bit 13 of the Network Configuration register is the pause enable control for reception. If this bit is set, transmission will pause if a non zero pause quantum frame is received.

If a valid pause frame is received then the Pause Time register is updated with the new frame's pause time, regardless of whether a previous pause frame is active or not. An interrupt (either bit 12 or bit 13 of the Interrupt Status register) is triggered when a pause frame is received, but only if the interrupt has been enabled (bit 12 and bit 13 of the Interrupt Mask register). Pause frames received with non zero quantum are indicated through the interrupt bit 12 of the Interrupt Status register. Pause frames received with zero quantum are indicated on bit 13 of the Interrupt Status register.

Once the Pause Time register is loaded and the frame currently being transmitted has been sent, no new frames are transmitted until the pause time reaches zero. The loading of a new pause time, and hence the pausing of

Downloaded from Arrow.com.

transmission, only occurs when the GMAC is configured for full duplex operation. If the GMAC is configured for half duplex there will be no transmission pause, but the pause frame received interrupt will still be triggered. A valid pause frame is defined as having a destination address that matches either the address stored in Specific Address register 1 or if it matches the reserved address of 0x0180C2000001. It must also have the MAC control frame type ID of 0x8808 and have the pause opcode of 0x0001.

Pause frames that have frame check sequence (FCS) or other errors will be treated as invalid and will be discarded. 802.3 Pause frames that are received after Priority-based Flow Control (PFC) has been negotiated will also be discarded. Valid pause frames received will increment the pause frames received statistic register.

The pause time register decrements every 512 bit times once transmission has stopped. For test purposes, the retry test bit can be set (bit 12 in the Network Configuration register) which causes the Pause Time register to decrement every GTXCK cycle once transmission has stopped.

The interrupt (bit 13 in the Interrupt Status register) is asserted whenever the Pause Time register decrements to zero (assuming it has been enabled by bit 13 in the Interrupt Mask register). This interrupt is also set when a zero quantum pause frame is received.

38.6.16.2 802.3 Pause Frame Transmission

Automatic transmission of pause frames is supported through the transmit pause frame bits of the Network Control register. If either bit 11 or bit 12 of the Network Control register is written with logic 1, an 802.3 pause frame will be transmitted, providing full duplex is selected in the Network Configuration register and the transmit block is enabled in the Network Control register.

Pause frame transmission will happen immediately if transmit is inactive or if transmit is active between the current frame and the next frame due to be transmitted.

Transmitted pause frames comprise the following:

- A destination address of 01-80-C2-00-00-01
- A source address taken from Specific Address register 1
- A type ID of 88-08 (MAC control frame)
- A pause opcode of 00-01
- A pause quantum register
- Fill of 00 to take the frame to minimum frame length
- Valid FCS

The pause quantum used in the generated frame will depend on the trigger source for the frame as follows:

- If bit 11 is written with a one, the pause quantum will be taken from the Transmit Pause Quantum register. The Transmit Pause Quantum register resets to a value of 0xFFFF giving maximum pause quantum as default.
- If bit 12 is written with a one, the pause quantum will be zero.

After transmission, a pause frame transmitted interrupt will be generated (bit 14 of the Interrupt Status register) and the only statistics register that will be incremented will be the Pause Frames Transmitted register.

Pause frames can also be transmitted by the MAC using normal frame transmission methods.

38.6.17 MAC PFC Priority-based Pause Frame Support

Note: Refer to the 802.1Qbb standard for a full description of priority-based pause operation.

The following table shows the start of a Priority-based Flow Control (PFC) pause frame.

Downloaded from Arrow.com.

Table 38-16. Start of a PFC Pause Frame

Address		Type (Mac Control Frame)	Pause Opcode	Priority Enable Vector	Pause Time
Destination	Source				
0x0180C2000001	6 bytes	0x8808	0x1001	2 bytes	8 × 2 bytes

The GMAC supports PFC priority-based pause transmission and reception. Before PFC pause frames can be received, bit 16 of the Network Control register must be set.

38.6.17.1 PFC Pause Frame Reception

The ability to receive and decode priority-based pause frames is enabled by setting bit 16 of the Network Control register. When this bit is set, the GMAC will match either classic 802.3 pause frames or PFC priority-based pause frames. Once a priority-based pause frame has been received and matched, then from that moment on the GMAC will only match on priority-based pause frames (this is an 802.1Qbb requirement, known as PFC negotiation). Once priority-based pause has been negotiated, any received 802.3x format pause frames will not be acted upon.

If a valid priority-based pause frame is received then the GMAC will decode the frame and determine which, if any, of the eight priorities require to be paused. Up to eight Pause Time registers are then updated with the eight pause times extracted from the frame regardless of whether a previous pause operation is active or not. An interrupt (either bit 12 or bit 13 of the Interrupt Status register) is triggered when a pause frame is received, but only if the interrupt has been enabled (bit 12 and bit 13 of the Interrupt Mask register). Pause frames received with non zero quantum are indicated through the interrupt bit 12 of the Interrupt Status register. Pause frames received with zero quantum are indicated on bit 13 of the Interrupt Status register. The loading of a new pause time only occurs when the GMAC is configured for full duplex operation. If the GMAC is configured for half duplex, the pause time counters will not be loaded, but the pause frame received interrupt will still be triggered. A valid pause frame is defined as having a destination address that matches either the address stored in Specific Address register 1 or if it matches the reserved address of 0x0180C2000001. It must also have the MAC control frame type ID of 0x8808 and have the pause opcode of 0x0101.

Pause frames that have frame check sequence (FCS) or other errors will be treated as invalid and will be discarded. Valid pause frames received will increment the Pause Frames Received Statistic register.

The Pause Time registers decrement every 512 bit times immediately following the PFC frame reception. For test purposes, the retry test bit can be set (bit 12 in the Network Configuration register) which causes the Pause Time register to decrement every GRXCK cycle once transmission has stopped.

The interrupt (bit 13 in the Interrupt Status register) is asserted whenever the Pause Time register decrements to zero (assuming it has been enabled by bit 13 in the Interrupt Mask register). This interrupt is also set when a zero quantum pause frame is received.

38.6.17.2 PFC Pause Frame Transmission

Automatic transmission of pause frames is supported through the transmit priority-based pause frame bit of the Network Control register. If bit 17 of the Network Control register is written with logic 1, a PFC pause frame will be transmitted providing full duplex is selected in the Network Configuration register and the transmit block is enabled in the Network Control register. When bit 17 of the Network Control register is set, the fields of the priority-based pause frame will be built using the values stored in the Transmit PFC Pause register.

Pause frame transmission will happen immediately if transmit is inactive or if transmit is active between the current frame and the next frame due to be transmitted.

Transmitted pause frames comprise the following:

- A destination address of 01-80-C2-00-00-01
- A source address taken from Specific Address register 1
- A type ID of 88-08 (MAC control frame)

Downloaded from Arrow.com.

- A pause opcode of 01-01
- A priority enable vector taken from Transmit PFC Pause register
- 8 pause quantum registers
- Fill of 00 to take the frame to minimum frame length
- Valid FCS

The pause quantum registers used in the generated frame will depend on the trigger source for the frame as follows:

- If bit 17 of the Network Control register is written with a one, then the priority enable vector of the priority-based pause frame will be set equal to the value stored in the Transmit PFC Pause register [7:0]. For each entry equal to zero in the Transmit PFC Pause register [15:8], the pause quantum field of the pause frame associated with that entry will be taken from the transmit pause quantum register. For each entry equal to one in the Transmit PFC Pause register [15:8], the pause quantum associated with that entry will be zero.
- The Transmit Pause Quantum register resets to a value of 0xFFFF giving maximum pause quantum as default.

After transmission, a pause frame transmitted interrupt will be generated (bit 14 of the Interrupt Status register) and the only statistics register that will be incremented will be the Pause Frames Transmitted register.

PFC Pause frames can also be transmitted by the MAC using normal frame transmission methods.

38.6.18 802.1Qav Support - Credit-based Shaping

A credit-based shaping algorithm is available on the two highest priority queues and is defined in the standard 802.1Qav: Forwarding and Queuing Enhancements for Time-Sensitive Streams. This allows traffic on these queues to be limited and to allow other queues to transmit.

Traffic shaping is enabled via the CBS (Credit Based Shaping) Control register. This enables a counter which stores the amount of transmit 'credit', measured in bytes that a particular queue has. A queue may only transmit if it has non-negative credit. If a queue has data to send, but is held off from doing as another queue is transmitting, then credit will accumulate in the credit counter at the rate defined in the IdleSlope register (GMAC_CBSISQx) for that queue. IdleSlope is the rate of change of credit when waiting to transmit and must be less than the value of the portTransmitRate. When this queue is transmitting the credit counter is decremented at the rate of sendSlope which is defined as the portTransmitRate - IdleSlope. A queue can accumulate negative credit when transmitting which will hold off any other transfers from that queue until credit returns to a non-negative value. No transfers are halted when a queue's credit becomes negative; it will accumulate negative credit until the transfer completes.

If both queues have positive credit, when the next queue to transfer is about to be selected, the queue with the most positive credit will be allowed to transfer first. The queue with the largest positive credit is the queue that had been prevented from transmitting for the longest time.

38.6.19 PHY Interface

Different PHY interfaces are supported by the Ethernet MAC:

- MII
- RMII

The MII interface is provided for 10/100 operation and uses txd[3:0] and rxd[3:0]. The RMII interface is provided for 10/100 operation and uses txd[1:0] and rxd[1:0].

38.6.20 10/100 Operation

The 10/100 Mbps speed bit in the Network Configuration register is used to select between 10 Mbps and 100 Mbps.

Downloaded from Arrow.com.

38.6.21 Jumbo Frames

The jumbo frames enable bit in the Network Configuration register allows the GMAC, in its default configuration, to receive jumbo frames up to 10240 bytes in size. This operation does not form part of the IEEE 802.3 specification and is normally disabled. When jumbo frames are enabled, frames received with a frame size greater than 10240 bytes are discarded.

38.7 Programming Interface

38.7.1 Initialization

38.7.1.1 Configuration

Initialization of the GMAC configuration (e.g., loop back mode, frequency ratios) must be done while the transmit and receive circuits are disabled. See the description of the Network Control register and Network Configuration register earlier in this document.

To change loop back mode, the following sequence of operations must be followed:

1. Write to Network Control register to disable transmit and receive circuits.
2. Write to Network Control register to change loop back mode.
3. Write to Network Control register to re-enable transmit or receive circuits.

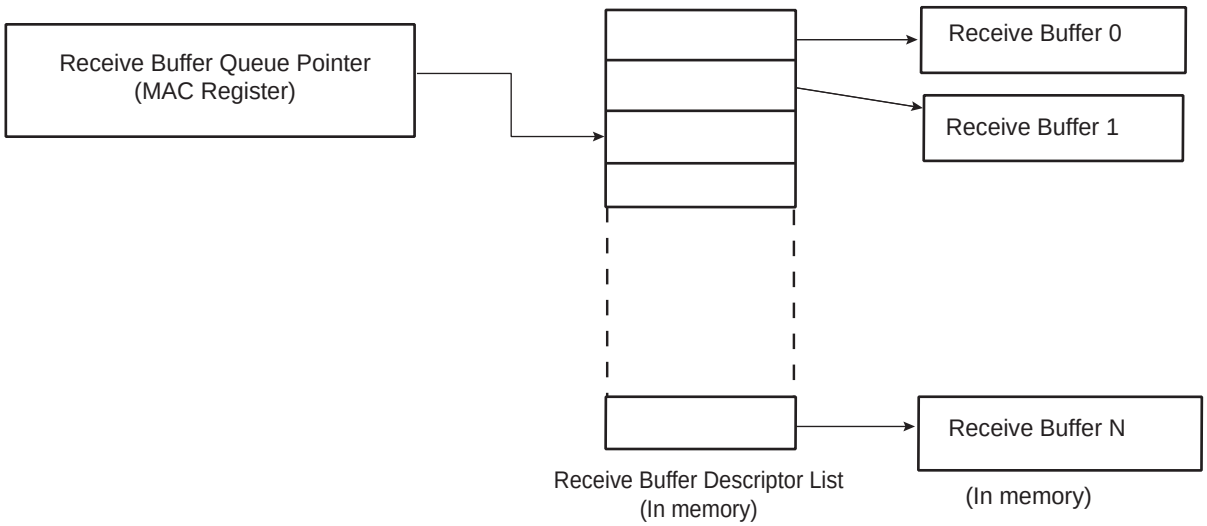
Note: These writes to the Network Control register cannot be combined in any way.

38.7.1.2 Receive Buffer List

Receive data is written to areas of data (i.e., buffers) in system memory. These buffers are listed in another data structure that also resides in main memory. This data structure (receive buffer queue) is a sequence of descriptor entries as defined in Table 38-4 “Receive Buffer Descriptor Entry”.

The Receive Buffer Queue Pointer register points to this data structure.

Figure 38-3. Receive Buffer List



To create the list of buffers:

1. Allocate a number (N) of buffers of X bytes in system memory, where X is the DMA buffer length programmed in the DMA Configuration register.
2. Allocate an area 8N bytes for the receive buffer descriptor list in system memory and create N entries in this list. Mark all entries in this list as owned by GMAC, i.e., bit 0 of word 0 set to 0.
3. Mark the last descriptor in the queue with the wrap bit (bit 1 in word 0 set to 1).

Downloaded from Arrow.com.

4. Write address of receive buffer descriptor list and control information to GMAC register receive buffer queue pointer
5. The receive circuits can then be enabled by writing to the address recognition registers and the Network Control register.

38.7.1.3 Transmit Buffer List

Transmit data is read from areas of data (the buffers) in system memory. These buffers are listed in another data structure that also resides in main memory. This data structure (Transmit Buffer Queue) is a sequence of descriptor entries as defined in Table 38-5 “Transmit Buffer Descriptor Entry”.

The Transmit Buffer Queue Pointer register points to this data structure.

To create this list of buffers:

1. Allocate a number (N) of buffers of between 1 and 2047 bytes of data to be transmitted in system memory. Up to 128 buffers per frame are allowed.
2. Allocate an area 8N bytes for the transmit buffer descriptor list in system memory and create N entries in this list. Mark all entries in this list as owned by GMAC, i.e., bit 31 of word 1 set to 0.
3. Mark the last descriptor in the queue with the wrap bit (bit 30 in word 1 set to 1).
4. Write address of transmit buffer descriptor list and control information to GMAC register transmit buffer queue pointer.
5. The transmit circuits can then be enabled by writing to the Network Control register.

38.7.1.4 Address Matching

The GMAC register pair hash address and the four Specific Address register pairs must be written with the required values. Each register pair comprises of a bottom register and top register, with the bottom register being written first. The address matching is disabled for a particular register pair after the bottom register has been written and re-enabled when the top register is written. Each register pair may be written at any time, regardless of whether the receive circuits are enabled or disabled.

As an example, to set Specific Address register 1 to recognize destination address 21:43:65:87:A9:CB, the following values are written to Specific Address register 1 bottom and Specific Address register 1 top:

- Specific Address register 1 bottom bits 31:0 (0x98): 0x8765_4321.
- Specific Address register 1 top bits 31:0 (0x9C): 0x0000_CBA9.

Note: The address matching is the first level of filtering. If there is a match, the screeners are the next level of filtering for routing the data to the appropriate queue. See Section 38.6.3.9 ”Priority Queueing in the DMA” for more details.

38.7.1.5 PHY Maintenance

The PHY Maintenance register is implemented as a shift register. Writing to the register starts a shift operation which is signalled as complete when bit two is set in the Network Status register (about 2000 MCK cycles later when bits 18:16 are set to 010 in the Network Configuration register). An interrupt is generated as this bit is set.

During this time, the MSB of the register is output on the MDIO pin and the LSB updated from the MDIO pin with each Management Data Clock (MDC) cycle. This causes the transmission of a PHY management frame on MDIO. See section 22.2.4.5 of the IEEE 802.3 standard.

Reading during the shift operation will return the current contents of the shift register. At the end of the management operation the bits will have shifted back to their original locations. For a read operation the data bits are updated with data read from the PHY. It is important to write the correct values to the register to ensure a valid PHY management frame is produced.

The Management Data Clock (MDC) should not toggle faster than 2.5 MHz (minimum period of 400 ns), as defined by the IEEE 802.3 standard. MDC is generated by dividing down MCK. Three bits in the Network Configuration register determine by how much MCK should be divided to produce MDC.

Downloaded from Arrow.com.

38.7.1.6 Interrupts

There are 18 interrupt conditions that are detected within the GMAC. The conditions are ORed to make multiple interrupts. Depending on the overall system design this may be passed through a further level of interrupt collection (interrupt controller). On receipt of the interrupt signal, the CPU enters the interrupt handler. Refer to the device interrupt controller documentation to identify that it is the GMAC that is generating the interrupt. To ascertain which interrupt, read the Interrupt Status register. Note that in the default configuration this register will clear itself after being read, though this may be configured to be write-one-to-clear if desired.

At reset all interrupts are disabled. To enable an interrupt, write to Interrupt Enable register with the pertinent interrupt bit set to 1. To disable an interrupt, write to Interrupt Disable register with the pertinent interrupt bit set to 1. To check whether an interrupt is enabled or disabled, read Interrupt Mask register. If the bit is set to 1, the interrupt is disabled.

38.7.1.7 Transmitting Frames

The procedure to set up a frame for transmission is the following:

1. Enable transmit in the Network Control register.
2. Allocate an area of system memory for transmit data. This does not have to be contiguous, varying byte lengths can be used if they conclude on byte borders.
3. Set-up the transmit buffer list by writing buffer addresses to word zero of the transmit buffer descriptor entries and control and length to word one.
4. Write data for transmission into the buffers pointed to by the descriptors.
5. Write the address of the first buffer descriptor to transmit buffer descriptor queue pointer.
6. Enable appropriate interrupts.
7. Write to the transmit start bit (TSTART) in the Network Control register.

38.7.1.8 Receiving Frames

When a frame is received and the receive circuits are enabled, the GMAC checks the address and, in the following cases, the frame is written to system memory:

- If it matches one of the four Specific Address registers.
- If it matches one of the four type ID registers.
- If it matches the hash address function.
- If it is a broadcast address (0xFFFFFFFFFFFF) and broadcasts are allowed.
- If the GMAC is configured to “copy all frames”.

The register receive buffer queue pointer points to the next entry in the receive buffer descriptor list and the GMAC uses this as the address in system memory to write the frame to.

Once the frame has been completely and successfully received and written to system memory, the GMAC then updates the receive buffer descriptor entry (see Table 38-4 “Receive Buffer Descriptor Entry”) with the reason for the address match and marks the area as being owned by software. Once this is complete, a receive complete interrupt is set. Software is then responsible for copying the data to the application area and releasing the buffer (by writing the ownership bit back to 0).

If the GMAC is unable to write the data at a rate to match the incoming frame, then a receive overrun interrupt is set. If there is no receive buffer available, i.e., the next buffer is still owned by software, a receive buffer not available interrupt is set. If the frame is not successfully received, a statistics register is incremented and the frame is discarded without informing software.

38.7.2 Statistics Registers

Statistics registers are described in the User Interface beginning with Section 38.8.47 ”GMAC Octets Transmitted Low Register” and ending with Section 38.8.91 ”GMAC UDP Checksum Errors Register”.

Downloaded from Arrow.com.

The statistics register block begins at 0x100 and runs to 0x1B0, and comprises the registers listed below.

Octets Transmitted Low Register
Octets Transmitted High Register
Frames Transmitted Register
Broadcast Frames Transmitted Register
Multicast Frames Transmitted Register
Pause Frames Transmitted Register
64 Byte Frames Transmitted Register
65 to 127 Byte Frames Transmitted Register
128 to 255 Byte Frames Transmitted Register
256 to 511 Byte Frames Transmitted Register
512 to 1023 Byte Frames Transmitted Register
1024 to 1518 Byte Frames Transmitted Register
Greater Than 1518 Byte Frames Transmitted Register
Transmit Underruns Register
Single Collision Frames Register
Multiple Collision Frames Register
Excessive Collisions Register
Late Collisions Register
Deferred Transmission Frames Register
Carrier Sense Errors Register
Octets Received Low Register
Octets Received High Register
Frames Received Register
Broadcast Frames Received Register
Multicast Frames Received Register
Pause Frames Received Register
64 Byte Frames Received Register
65 to 127 Byte Frames Received Register
128 to 255 Byte Frames Received Register
256 to 511 Byte Frames Received Register
512 to 1023 Byte Frames Received Register
1024 to 1518 Byte Frames Received Register
1519 to Maximum Byte Frames Received Register
Undersize Frames Received Register
Oversize Frames Received Register
Jabbers Received Register
Frame Check Sequence Errors Register
Length Field Frame Errors Register
Receive Symbol Errors Register
Alignment Errors Register
Receive Resource Errors Register
Receive Overrun Register
IP Header Checksum Errors Register
TCP Checksum Errors Register
UDP Checksum Errors Register

These registers reset to zero on a read and stick at all ones when they count to their maximum value. They should be read frequently enough to prevent loss of data.

The receive statistics registers are only incremented when the receive enable bit (RXEN) is set in the Network Control register.

Once a statistics register has been read, it is automatically cleared. When reading the Octets Transmitted and Octets Received registers, bits 31:0 should be read prior to bits 47:32 to ensure reliable operation.

Downloaded from Arrow.com.

38.8 Ethernet MAC (GMAC) User Interface

Table 38-17. Register Mapping

Offset[1] [2]	Register	Name	Access	Reset
0x000	Network Control Register	GMAC_NCR	Read/Write	0x0000_0000
0x004	Network Configuration Register	GMAC_NCFGR	Read/Write	0x0008_0000
0x008	Network Status Register	GMAC_NSR	Read-only	0b01x0
0x00C	User Register	GMAC_UR	Read/Write	0x0000_0000
0x010	DMA Configuration Register	GMAC_DCFGR	Read/Write	0x0002_0004
0x014	Transmit Status Register	GMAC_TSR	Read/Write	0x0000_0000
0x018	Receive Buffer Queue Base Address Register	GMAC_RBQB	Read/Write	0x0000_0000
0x01C	Transmit Buffer Queue Base Address Register	GMAC_TBQB	Read/Write	0x0000_0000
0x020	Receive Status Register	GMAC_RSR	Read/Write	0x0000_0000
0x024	Interrupt Status Register	GMAC_ISR	Read-only	0x0000_0000
0x028	Interrupt Enable Register	GMAC_IER	Write-only	–
0x02C	Interrupt Disable Register	GMAC_IDR	Write-only	–
0x030	Interrupt Mask Register	GMAC_IMR	Read/Write	0x07FF_FFFF
0x034	PHY Maintenance Register	GMAC_MAN	Read/Write	0x0000_0000
0x038	Received Pause Quantum Register	GMAC_RPQ	Read-only	0x0000_0000
0x03C	Transmit Pause Quantum Register	GMAC_TPQ	Read/Write	0x0000_FFFF
0x040	TX Partial Store and Forward Register	GMAC_TPSF	Read/Write	0x0000_0FFF
0x044	RX Partial Store and Forward Register	GMAC_RPSF	Read/Write	0x0000_0FFF
0x048	RX Jumbo Frame Max Length Register	GMAC_RJFML	Read/Write	0x0000_3FFF
0x4C–0x07C	Reserved	–	–	–
0x080	Hash Register Bottom	GMAC_HRB	Read/Write	0x0000_0000
0x084	Hash Register Top	GMAC_HRT	Read/Write	0x0000_0000
0x088	Specific Address 1 Bottom Register	GMAC_SAB1	Read/Write	0x0000_0000
0x08C	Specific Address 1 Top Register	GMAC_SAT1	Read/Write	0x0000_0000
0x090	Specific Address 2 Bottom Register	GMAC_SAB2	Read/Write	0x0000_0000
0x094	Specific Address 2 Top Register	GMAC_SAT2	Read/Write	0x0000_0000
0x098	Specific Address 3 Bottom Register	GMAC_SAB3	Read/Write	0x0000_0000
0x09C	Specific Address 3 Top Register	GMAC_SAT3	Read/Write	0x0000_0000
0x0A0	Specific Address 4 Bottom Register	GMAC_SAB4	Read/Write	0x0000_0000
0x0A4	Specific Address 4 Top Register	GMAC_SAT4	Read/Write	0x0000_0000
0x0A8	Type ID Match 1 Register	GMAC_TIDM1	Read/Write	0x0000_0000
0x0AC	Type ID Match 2 Register	GMAC_TIDM2	Read/Write	0x0000_0000
0x0B0	Type ID Match 3 Register	GMAC_TIDM3	Read/Write	0x0000_0000
0x0B4	Type ID Match 4 Register	GMAC_TIDM4	Read/Write	0x0000_0000
0x0B8	Wake on LAN Register	GMAC_WOL	Read/Write	0x0000_0000

Downloaded from Arrow.com.

Table 38-17. Register Mapping (Continued)

Offset[1] [2]	Register	Name	Access	Reset
0x0BC	IPG Stretch Register	GMAC_IPGS	Read/Write	0x0000_0000
0x0C0	Stacked VLAN Register	GMAC_SVLAN	Read/Write	0x0000_0000
0x0C4	Transmit PFC Pause Register	GMAC_TPFCP	Read/Write	0x0000_0000
0x0C8	Specific Address 1 Mask Bottom Register	GMAC_SAMB1	Read/Write	0x0000_0000
0x0CC	Specific Address 1 Mask Top Register	GMAC_SAMT1	Read/Write	0x0000_0000
0x0D0–0x0D8	Reserved	–	–	–
0x0DC	1588 Timer Nanosecond Comparison Register	GMAC_NSC	Read/Write	0x0000_0000
0x0E0	1588 Timer Second Comparison Low Register	GMAC_SCL	Read/Write	0x0000_0000
0x0E4	1588 Timer Second Comparison High Register	GMAC_SCH	Read/Write	0x0000_0000
0x0E8	PTP Event Frame Transmitted Seconds High Register	GMAC_EFTSH	Read-only	0x0000_0000
0x0EC	PTP Event Frame Received Seconds High Register	GMAC_EFRSH	Read-only	0x0000_0000
0x0F0	PTP Peer Event Frame Transmitted Seconds High Register	GMAC_PEFTSH	Read-only	0x0000_0000
0x0F4	PTP Peer Event Frame Received Seconds High Register	GMAC_PEFRSH	Read-only	0x0000_0000
0x0E8–0x0FC	Reserved	–	–	–
0x100	Octets Transmitted Low Register	GMAC_OTLO	Read-only	0x0000_0000
0x104	Octets Transmitted High Register	GMAC_OTHI	Read-only	0x0000_0000
0x108	Frames Transmitted Register	GMAC_FT	Read-only	0x0000_0000
0x10C	Broadcast Frames Transmitted Register	GMAC_BCFT	Read-only	0x0000_0000
0x110	Multicast Frames Transmitted Register	GMAC_MFT	Read-only	0x0000_0000
0x114	Pause Frames Transmitted Register	GMAC_PFT	Read-only	0x0000_0000
0x118	64 Byte Frames Transmitted Register	GMAC_BFT64	Read-only	0x0000_0000
0x11C	65 to 127 Byte Frames Transmitted Register	GMAC_TBFT127	Read-only	0x0000_0000
0x120	128 to 255 Byte Frames Transmitted Register	GMAC_TBFT255	Read-only	0x0000_0000
0x124	256 to 511 Byte Frames Transmitted Register	GMAC_TBFT511	Read-only	0x0000_0000
0x128	512 to 1023 Byte Frames Transmitted Register	GMAC_TBFT1023	Read-only	0x0000_0000
0x12C	1024 to 1518 Byte Frames Transmitted Register	GMAC_TBFT1518	Read-only	0x0000_0000
0x130	Greater Than 1518 Byte Frames Transmitted Register	GMAC_GTBFT1518	Read-only	0x0000_0000
0x134	Transmit Underruns Register	GMAC_TUR	Read-only	0x0000_0000
0x138	Single Collision Frames Register	GMAC_SCF	Read-only	0x0000_0000
0x13C	Multiple Collision Frames Register	GMAC_MCF	Read-only	0x0000_0000
0x140	Excessive Collisions Register	GMAC_EC	Read-only	0x0000_0000
0x144	Late Collisions Register	GMAC_LC	Read-only	0x0000_0000
0x148	Deferred Transmission Frames Register	GMAC_DTF	Read-only	0x0000_0000
0x14C	Carrier Sense Errors Register	GMAC_CSE	Read-only	0x0000_0000

Downloaded from Arrow.com.

Table 38-17. Register Mapping (Continued)

Offset[1] [2]	Register	Name	Access	Reset
0x150	Octets Received Low Received Register	GMAC_ORLO	Read-only	0x0000_0000
0x154	Octets Received High Received Register	GMAC_ORHI	Read-only	0x0000_0000
0x158	Frames Received Register	GMAC_FR	Read-only	0x0000_0000
0x15C	Broadcast Frames Received Register	GMAC_BCFR	Read-only	0x0000_0000
0x160	Multicast Frames Received Register	GMAC_MFR	Read-only	0x0000_0000
0x164	Pause Frames Received Register	GMAC_PFR	Read-only	0x0000_0000
0x168	64 Byte Frames Received Register	GMAC_BFR64	Read-only	0x0000_0000
0x16C	65 to 127 Byte Frames Received Register	GMAC_TBFR127	Read-only	0x0000_0000
0x170	128 to 255 Byte Frames Received Register	GMAC_TBFR255	Read-only	0x0000_0000
0x174	256 to 511 Byte Frames Received Register	GMAC_TBFR511	Read-only	0x0000_0000
0x178	512 to 1023 Byte Frames Received Register	GMAC_TBFR1023	Read-only	0x0000_0000
0x17C	1024 to 1518 Byte Frames Received Register	GMAC_TBFR1518	Read-only	0x0000_0000
0x180	1519 to Maximum Byte Frames Received Register	GMAC_TMXBFR	Read-only	0x0000_0000
0x184	Undersize Frames Received Register	GMAC_UFR	Read-only	0x0000_0000
0x188	Oversize Frames Received Register	GMAC_OFR	Read-only	0x0000_0000
0x18C	Jabbers Received Register	GMAC_JR	Read-only	0x0000_0000
0x190	Frame Check Sequence Errors Register	GMAC_FCSE	Read-only	0x0000_0000
0x194	Length Field Frame Errors Register	GMAC_LFFE	Read-only	0x0000_0000
0x198	Receive Symbol Errors Register	GMAC_RSE	Read-only	0x0000_0000
0x19C	Alignment Errors Register	GMAC_AE	Read-only	0x0000_0000
0x1A0	Receive Resource Errors Register	GMAC_RRE	Read-only	0x0000_0000
0x1A4	Receive Overrun Register	GMAC_ROE	Read-only	0x0000_0000
0x1A8	IP Header Checksum Errors Register	GMAC_IHCE	Read-only	0x0000_0000
0x1AC	TCP Checksum Errors Register	GMAC_TCE	Read-only	0x0000_0000
0x1B0	UDP Checksum Errors Register	GMAC_UCE	Read-only	0x0000_0000
0x1B4–0x1B8	Reserved	–	–	–
0x1BC	1588 Timer Increment Sub-nanoseconds Register	GMAC_TISUBN	Read/Write	0x0000_0000
0x1C0	1588 Timer Seconds High Register	GMAC_TSH	Read/Write	0x0000_0000
0x1C4–0x1CC	Reserved	–	–	–
0x1D0	1588 Timer Seconds Low Register	GMAC_TSL	Read/Write	0x0000_0000
0x1D4	1588 Timer Nanoseconds Register	GMAC_TN	Read/Write	0x0000_0000
0x1D8	1588 Timer Adjust Register	GMAC_TA	Write-only	–
0x1DC	1588 Timer Increment Register	GMAC_TI	Read/Write	0x0000_0000
0x1E0	PTP Event Frame Transmitted Seconds Low Register	GMAC_EFTSL	Read-only	0x0000_0000
0x1E4	PTP Event Frame Transmitted Nanoseconds Register	GMAC_EFTN	Read-only	0x0000_0000
0x1E8	PTP Event Frame Received Seconds Low Register	GMAC_EFRSL	Read-only	0x0000_0000

Downloaded from Arrow.com.

Table 38-17. Register Mapping (Continued)

Offset[1] [2]	Register	Name	Access	Reset
0x1EC	PTP Event Frame Received Nanoseconds Register	GMAC_EFRN	Read-only	0x0000_0000
0x1F0	PTP Peer Event Frame Transmitted Seconds Low Register	GMAC_PEFTSL	Read-only	0x0000_0000
0x1F4	PTP Peer Event Frame Transmitted Nanoseconds Register	GMAC_PEFTN	Read-only	0x0000_0000
0x1F8	PTP Peer Event Frame Received Seconds Low Register	GMAC_PEFRSL	Read-only	0x0000_0000
0x1FC	PTP Peer Event Frame Received Nanoseconds Register	GMAC_PEFRN	Read-only	0x0000_0000
0x200–0x3FC	Reserved	–	–	–
0x3FC + (index * 0x04)	Interrupt Status Register Priority Queue [3]	GMAC_ISRPQ	Read-only	0x0000_0000
0x43C + (index * 0x04)	Transmit Buffer Queue Base Address Register Priority Queue [3]	GMAC_TBQBAPQ	Read/Write	0x0000_0000
0x47C + (index * 0x04)	Receive Buffer Queue Base Address Register Priority Queue [3]	GMAC_RBQBAPQ	Read/Write	0x0000_0000
0x49C + (index * 0x04)	Receive Buffer Size Register Priority Queue [3]	GMAC_RBSRPQ	Read/Write	0x0000_0002
0x4BC	Credit-Based Shaping Control Register	GMAC_CBSCR	Read/Write	0x0000_0000
0x4C0	Credit-Based Shaping IdleSlope Register for Queue A	GMAC_CBSISQA	Read/Write	0x0000_0000
0x4C4	Credit-Based Shaping IdleSlope Register for Queue B	GMAC_CBSISQB	Read/Write	0x0000_0000
0x500 + (index * 0x04)	Screening Type 1 Register Priority Queue [4]	GMAC_ST1RPQ	Read/Write	0x0000_0000
0x540 + (index * 0x04)	Screening Type 2 Register Priority Queue [5]	GMAC_ST2RPQ	Read/Write	0x0000_0000
0x5FC + (index * 0x04)	Interrupt Enable Register Priority Queue [3]	GMAC_IERPQ	Write-only	–
0x61C + (index * 0x04)	Interrupt Disable Register Priority Queue [3]	GMAC_IDRPQ	Write-only	–
0x63C + (index * 0x04)	Interrupt Mask Register Priority Queue [3]	GMAC_IMRPQ	Read/Write	0x0000_0000
0x6E0 + (index * 0x04)	Screening Type 2 Ethertype Register [6]	GMAC_ST2ER	Read/Write	0x0000_0000
0x700 + (index * 0x08)	Screening Type 2 Compare Word 0 Register [7]	GMAC_ST2CW0	Read/Write	0x0000_0000
0x704 + (index * 0x08)	Screening Type 2 Compare Word 1 Register [7]	GMAC_ST2CW1	Read/Write	0x0000_0000

Notes:
1. If an offset is not listed in the Register Mapping, it must be considered as ‘reserved’.
2. Some register groups are not continuous in memory.
3. The index range for the following registers is from 1 to 2:
   - GMAC_ISRPQ
   - GMAC_TBQBAPQ
   - GMAC_RBQBAPQ
   - GMAC_RBSRPQ
   - GMAC_IERPQ
   - GMAC_IDRPQ
   - GMAC_IMRPQ
4. The index for GMAC_ST1RPQ registers ranges from 0 to 3.
5. The index for GMAC_ST2RPQ registers ranges from 0 to 7.
6. The index for GMAC_ST2ER registers ranges from 0 to 3.
7. The index for GMAC_ST2CW0 and GMAC_ST2CW1 registers ranges from 0 to 23.

Downloaded from Arrow.com.

38.8.1 GMAC Network Control Register

Name: GMAC_NCR

Address: 0x40050000

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	FNP	TXPBPF	ENPBPR
15	14	13	12	11	10	9	8
SRTSM	–	–	TXZQPF	TXPF	THALT	TSTART	BP
7	6	5	4	3	2	1	0
WESTAT	INCSTAT	CLRSTAT	MPE	TXEN	RXEN	LBL	–

- **LBL: Loop Back Local**

Connects GTX to GRX，GTXEN to GRXDV and forces full duplex mode. GRXCK and GTXCK may malfunction as the GMAC is switched into and out of internal loop back. It is important that receive and transmit circuits have already been disabled when making the switch into and out of internal loop back.

- **RXEN: Receive Enable**

When set, RXEN enables the GMAC to receive data. When reset frame reception stops immediately and the receive pipeline will be cleared. The Receive Queue Pointer Register is unaffected.

- **TXEN: Transmit Enable**

When set, TXEN enables the GMAC transmitter to send data. When reset transmission will stop immediately, the transmit pipeline and control registers will be cleared and the Transmit Queue Pointer Register will reset to point to the start of the transmit descriptor list.

- **MPE: Management Port Enable**

Set to one to enable the management port. When zero, forces MDIO to high impedance state and MDC low.

- **CLRSTAT: Clear Statistics Registers**

This bit is write-only. Writing a one clears the statistics registers.

- **INCSTAT: Increment Statistics Registers**

This bit is write-only. Writing a one increments all the statistics registers by one for test purposes.

- **WESTAT: Write Enable for Statistics Registers**

Setting this bit to one makes the statistics registers writable for functional test purposes.

- **BP: Back pressure**

If set in 10M or 100M half duplex mode, forces collisions on all received frames.

- **TSTART: Start Transmission**

Writing one to this bit starts transmission.

Downloaded from Arrow.com.

- **THALT: Transmit Halt**

Writing one to this bit halts transmission as soon as any ongoing frame transmission ends.

- **TXPF: Transmit Pause Frame**

Writing one to this bit causes a pause frame to be transmitted.

- **TXZQPF: Transmit Zero Quantum Pause Frame**

Writing one to this bit causes a pause frame with zero quantum to be transmitted.

- **SRTSM: Store Receive Time Stamp to Memory**

0: Normal operation.

1: Causes the CRC of every received frame to be replaced with the value of the nanoseconds field of the 1588 timer that was captured as the receive frame passed the message time stamp point.

- **ENPBPR: Enable PFC Priority-based Pause Reception**

Enables PFC Priority Based Pause Reception capabilities. Setting this bit enables PFC negotiation and recognition of priority-based pause frames.

- **TXPBPF: Transmit PFC Priority-based Pause Frame**

Takes the values stored in the Transmit PFC Pause Register.

- **FNP: Flush Next Packet**

Flush the next packet from the external RX DPRAM. Writing one to this bit will only have an effect if the DMA is not currently writing a packet already stored in the DPRAM to memory.

Downloaded from Arrow.com.

38.8.2 GMAC Network Configuration Register

Name: GMAC_NCFGR

Address: 0x40050004

Access: Read/Write

31	30	29	28	27	26	25	24
–	IRXER	RXBP	IPGSEN	–	IRXFCS	EFRHD	RXCOEN
23	22	21	20	19	18	17	16
DCPF	DBW		CLK			RFCS	LFERD
15	14	13	12	11	10	9	8
RXBUFO		PEN	RTY	–	–	–	MAXFS
7	6	5	4	3	2	1	0
UNIHEN	MTI HEN	NBC	CAF	JFRAME	DNVLAN	FD	SPD

- **SPD: Speed**

Set to logic one to indicate 100 Mbps operation, logic zero for 10 Mbps.

- **FD: Full Duplex**

If set to logic one, the transmit block ignores the state of collision and carrier sense and allows receive while transmitting.

- **DNVLAN: Discard Non-VLAN FRAMES**

When set only VLAN tagged frames will be passed to the address matching logic.

- **JFRAME: Jumbo Frame Size**

Set to one to enable jumbo frames up to 10240 bytes to be accepted. The default length is 10240 bytes.

- **CAF: Copy All Frames**

When set to logic one, all valid frames will be accepted.

- **NBC: No Broadcast**

When set to logic one, frames addressed to the broadcast address of all ones will not be accepted.

- **MTIHEN: Multicast Hash Enable**

When set, multicast frames will be accepted when the 6-bit hash function of the destination address points to a bit that is set in the Hash Register.

- **UNIHEN: Unicast Hash Enable**

When set, unicast frames will be accepted when the 6-bit hash function of the destination address points to a bit that is set in the Hash Register.

- **MAXFS: 1536 Maximum Frame Size**

Setting this bit means the GMAC will accept frames up to 1536 bytes in length. Normally the GMAC would reject any frame above 1518 bytes.

- **RTY: Retry Test**

Downloaded from Arrow.com.

Must be set to zero for normal operation. If set to one the backoff between collisions will always be one slot time. Setting this bit to one helps test the too many retries condition. Also used in the pause frame tests to reduce the pause counter's decrement time from 512 bit times, to every GRXCK cycle.

- **PEN: Pause Enable**

When set, transmission will pause if a non-zero 802.3 classic pause frame is received and PFC has not been negotiated.

- **RXBUFO: Receive Buffer Offset**

Indicates the number of bytes by which the received data is offset from the start of the receive buffer

- **LFERD: Length Field Error Frame Discard**

Setting this bit causes frames with a measured length shorter than the extracted length field (as indicated by bytes 13 and 14 in a non-VLAN tagged frame) to be discarded. This only applies to frames with a length field less than 0x0600.

- **RFCS: Remove FCS**

Setting this bit will cause received frames to be written to memory without their frame check sequence (last 4 bytes). The frame length indicated will be reduced by four bytes in this mode.

- **CLK: MDC CLock Division**

Set according to MCK speed. These three bits determine the number MCK will be divided by to generate Management Data Clock (MDC). For conformance with the 802.3 specification, MDC must not exceed 2.5 MHz (MDC is only active during MDIO read and write operations).

Value	Name	Description
0	MCK_8	MCK divided by 8 (MCK up to 20 MHz)
1	MCK_16	MCK divided by 16 (MCK up to 40 MHz)
2	MCK_32	MCK divided by 32 (MCK up to 80 MHz)
3	MCK_48	MCK divided by 48 (MCK up to 120 MHz)
4	MCK_64	MCK divided by 64 (MCK up to 160 MHz)
5	MCK_96	MCK divided by 96 (MCK up to 240 MHz)

- **DBW: Data Bus Width**

Should always be written to 0.

- **DCPF: Disable Copy of Pause Frames**

Set to one to prevent valid pause frames being copied to memory. When set, pause frames are not copied to memory regardless of the state of the Copy All Frames bit, whether a hash match is found or whether a type ID match is identified. If a destination address match is found, the pause frame will be copied to memory. Note that valid pause frames received will still increment pause statistics and pause the transmission of frames as required.

- **RXCOEN: Receive Checksum Offload Enable**

When set, the receive checksum engine is enabled. Frames with bad IP, TCP or UDP checksums are discarded.

- **EFRHD: Enable Frames Received in Half Duplex**

Enable frames to be received in half-duplex mode while transmitting.

- **IRXFCS: Ignore RX FCS**

Downloaded from Arrow.com.

When set, frames with FCS/CRC errors will not be rejected. FCS error statistics will still be collected for frames with bad FCS and FCS status will be recorded in frame’s DMA descriptor. For normal operation this bit must be set to zero.

- **IPGSEN: IP Stretch Enable**

When set, the transmit IPG can be increased above 96 bit times depending on the previous frame length using the IPG Stretch Register.

- **RXBP: Receive Bad Preamble**

When set, frames with non-standard preamble are not rejected.

- **IRXER: Ignore IPG GRXER**

When set, GRXER has no effect on the GMAC's operation when GRXDV is low.

Downloaded from Arrow.com.

38.8.3 GMAC Network Status Register

Name: GMAC_NSR

Address: 0x40050008

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	IDLE	MDIO	–

- **MDIO: MDIO Input Status**

Returns status of the MDIO pin.

- **IDLE: PHY Management Logic Idle**

The PHY management logic is idle (i.e., has completed).

Downloaded from Arrow.com.

38.8.4 GMAC User Register

Name: GMAC_UR

Address: 0x4005000C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	RMII

- **RMII: Reduced MII Mode**

0: RMII mode is selected (default).

1: MII mode is selected.

Downloaded from Arrow.com.

38.8.5 GMAC DMA Configuration Register

Name: GMAC_DCFGR

Address: 0x40050010

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	DDRP

23	22	21	20	19	18	17	16
DRBS							

15	14	13	12	11	10	9	8
–	–	–	–	TXCOEN	TXPBMS	RXBMS	

7	6	5	4	3	2	1	0
ESPA	ESMA	–	FBLDO				

- **FBLDO: Fixed Burst Length for DMA Data Operations:**

Selects the burst length to attempt to use on the AHB when transferring frame data. Not used for DMA management operations and only used where space and data size allow. Otherwise SINGLE type AHB transfers are used.

One-hot priority encoding enforced automatically on register writes as follows, where ‘x’ represents don’t care:

Value	Name	Description
0	–	Reserved
1	SINGLE	00001: Always use SINGLE AHB bursts
2	–	Reserved
4	INCR4	001xx: Attempt to use INCR4 AHB bursts (Default)
8	INCR8	01xxx: Attempt to use INCR8 AHB bursts
16	INCR16	1xxxx: Attempt to use INCR16 AHB bursts

- **ESMA: Endian Swap Mode Enable for Management Descriptor Accesses**

When set, selects swapped endianism for AHB transfers. When clear, selects little endian mode.

- **ESPA: Endian Swap Mode Enable for Packet Data Accesses**

When set, selects swapped endianism for AHB transfers. When clear, selects little endian mode.

- **RXBMS: Receiver Packet Buffer Memory Size Select**

The default receive packet buffer size is 4 Kbytes. The table below shows how to configure this memory to FULL, HALF, QUARTER or EIGHTH of the default size.

Value	Name	Description
0	EIGHTH	4/8 Kbyte Memory Size
1	QUARTER	4/4 Kbytes Memory Size
2	HALF	4/2 Kbytes Memory Size
3	FULL	4 Kbytes Memory Size

- **TXPBMS: Transmitter Packet Buffer Memory Size Select**

Downloaded from Arrow.com.

Having this bit at zero halves the amount of memory used for the transmit packet buffer. This reduces the amount of memory used by the GMAC. It is important to set this bit to one if the full configured physical memory is available. The value in brackets below represents the size that would result for the default maximum configured memory size of 4 Kbytes.

0: Do not use top address bit (2 Kbytes).

1: Use full configured addressable space (4 Kbytes).

- **TXCOEN: Transmitter Checksum Generation Offload Enable**

Transmitter IP, TCP and UDP checksum generation offload enable. When set, the transmitter checksum generation engine is enabled to calculate and substitute checksums for transmit frames. When clear, frame data is unaffected.

- **DRBS: DMA Receive Buffer Size**

DMA receive buffer size in AHB system memory. The value defined by these bits determines the size of buffer to use in main AHB system memory when writing received data.

The value is defined in multiples of 64 bytes, thus a value of 0x01 corresponds to buffers of 64 bytes, 0x02 corresponds to 128 bytes etc.

For example:

– 0x02: 128 bytes

– 0x18: 1536 bytes (1 × max length frame/buffer)

– 0xA0: 10240 bytes (1 × 10K jumbo frame/buffer)

Note that this value should never be written as zero.

- **DDRP: DMA Discard Receive Packets**

When set, the GMAC DMA will automatically discard receive packets from the receiver packet buffer memory when no AHB resource is available.

When low, the received packets will remain to be stored in the SRAM based packet buffer until AHB buffer resource next becomes available.

A write to this bit is ignored if the DMA is not configured in the packet buffer full store and forward mode.

Downloaded from Arrow.com.

38.8.6 GMAC Transmit Status Register

Name: GMAC_TSR

Address: 0x40050014

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	HRESP
7	6	5	4	3	2	1	0
–	–	TXCOMP	TFC	TXGO	RLE	COL	UBR

- **UBR: Used Bit Read**

Set when a transmit buffer descriptor is read with its used bit set. Writing a one clears this bit.

- **COL: Collision Occurred**

Set by the assertion of collision. Writing a one clears this bit. When operating in 10/100 mode, this status indicates either a collision or a late collision.

- **RLE: Retry Limit Exceeded**

Writing a one clears this bit.

- **TXGO: Transmit Go**

Transmit go, if high transmit is active. When using the DMA interface this bit represents the TXGO variable as specified in the transmit buffer description.

- **TFC: Transmit Frame Corruption Due to AHB Error**

Transmit frame corruption due to AHB error. Set if an error occurs while midway through reading transmit frame from the AHB, including HRESP errors and buffers exhausted mid frame (if the buffers run out during transmission of a frame then transmission stops, FCS shall be bad and GTXER asserted).

Also set in DMA packet buffer mode if single frame is too large for configured packet buffer memory size.

Writing a one clears this bit.

- **TXCOMP: Transmit Complete**

Set when a frame has been transmitted. Writing a one clears this bit.

- **HRESP: HRESP Not OK**

Set when the DMA block sees HRESP not OK. Writing a one clears this bit.

Downloaded from Arrow.com.

38.8.7 GMAC Receive Buffer Queue Base Address Register

Name: GMAC_RBQB

Address: 0x40050018

Access: Read/Write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR						–	–

This register holds the start address of the receive buffer queue (receive buffers descriptor list). The receive buffer queue base address must be initialized before receive is enabled through bit 2 of the Network Control Register. Once reception is enabled, any write to the Receive Buffer Queue Base Address Register is ignored. Reading this register returns the location of the descriptor currently being accessed. This value increments as buffers are used. Software should not use this register for determining where to remove received frames from the queue as it constantly changes as new frames are received. Software should instead work its way through the buffer descriptor queue checking the “used” bits.

In terms of AMBA AHB operation, the descriptors are read from memory using a single 32-bit AHB access. The descriptors should be aligned at 32-bit boundaries and the descriptors are written to using two individual non sequential accesses.

- **ADDR: Receive Buffer Queue Base Address**

Written with the address of the start of the receive queue.

Downloaded from Arrow.com.

38.8.8 GMAC Transmit Buffer Queue Base Address Register

Name: GMAC_TBQB

Address: 0x4005001C

Access: Read/Write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR						–	–

This register holds the start address of the transmit buffer queue (transmit buffers descriptor list). The Transmit Buffer Queue Base Address Register must be initialized before transmit is started through bit 9 of the Network Control Register. Once transmission has started, any write to the Transmit Buffer Queue Base Address Register is illegal and therefore ignored.

Note that due to clock boundary synchronization, it takes a maximum of four MCK cycles from the writing of the transmit start bit before the transmitter is active. Writing to the Transmit Buffer Queue Base Address Register during this time may produce unpredictable results.

Reading this register returns the location of the descriptor currently being accessed. Since the DMA handles two frames at once, this may not necessarily be pointing to the current frame being transmitted.

In terms of AMBA AHB operation, the descriptors are written to memory using a single 32-bit AHB access. The descriptors should be aligned at 32-bit boundaries and the descriptors are read from memory using two individual non sequential accesses.

- **ADDR: Transmit Buffer Queue Base Address**

Written with the address of the start of the transmit queue.

Downloaded from Arrow.com.

38.8.9 GMAC Receive Status Register

Name: GMAC_RSR

Address: 0x40050020

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	HNO	RXOVR	REC	BNA

This register, when read, provides receive status details. Once read, individual bits may be cleared by writing a one to them. It is not possible to set a bit to 1 by writing to the register.

- **BNA: Buffer Not Available**

An attempt was made to get a new buffer and the pointer indicated that it was owned by the processor. The DMA will re-read the pointer each time an end of frame is received until a valid pointer is found. This bit is set following each descriptor read attempt that fails, even if consecutive pointers are unsuccessful and software has in the mean time cleared the status flag. Writing a one clears this bit.

- **REC: Frame Received**

One or more frames have been received and placed in memory. Writing a one clears this bit.

- **RXOVR: Receive Overrun**

This bit is set if the receive status was not taken at the end of the frame. This bit is also set if the packet buffer overflows. The buffer will be recovered if an overrun occurs. Writing a one clears this bit.

- **HNO: HRESP Not OK**

Set when the DMA block sees HRESP not OK. Writing a one clears this bit.

Downloaded from Arrow.com.

38.8.10 GMAC Interrupt Status Register

Name: GMAC_ISR

Address: 0x40050024

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	WOL	–	SRI	PDRSFT	PDRQFT
23	22	21	20	19	18	17	16
PDRSFR	PDRQFR	SFT	DRQFT	SFR	DRQFR	–	–
15	14	13	12	11	10	9	8
–	PFTR	PTZ	PFNZ	HRESP	ROVR	–	–
7	6	5	4	3	2	1	0
TCOMP	TFC	RLEX	TUR	TXUBR	RXUBR	RCOMP	MFS

This register indicates the source of the interrupt. In order that the bits of this register read 1, the corresponding interrupt source must be enabled in the mask register. If any bit is set in this register, the GMAC interrupt signal will be asserted in the system.

- **MFS: Management Frame Sent**

The PHY Maintenance Register has completed its operation. Cleared on read.

- **RCOMP: Receive Complete**

A frame has been stored in memory. Cleared on read.

- **RXUBR: RX Used Bit Read**

Set when a receive buffer descriptor is read with its used bit set. Cleared on read.

- **TXUBR: TX Used Bit Read**

Set when a transmit buffer descriptor is read with its used bit set. Cleared on read.

- **TUR: Transmit Underrun**

This interrupt is set if the transmitter was forced to terminate a frame that it has already began transmitting due to further data being unavailable.

This interrupt is set if a transmitter status write back has not completed when another status write back is attempted.

This interrupt is also set when the transmit DMA has written the SOP data into the FIFO and either the AHB bus was not granted in time for further data, or because an AHB not OK response was returned, or because the used bit was read.

- **RLEX: Retry Limit Exceeded**

Transmit error. Cleared on read.

- **TFC: Transmit Frame Corruption Due to AHB Error**

Transmit frame corruption due to AHB error. Set if an error occurs while midway through reading transmit frame from the AHB, including HRESP errors and buffers exhausted mid frame.

- **TCOMP: Transmit Complete**

Set when a frame has been transmitted. Cleared on read.

Downloaded from Arrow.com.

- **ROVR: Receive Overrun**

Set when the receive overrun status bit is set. Cleared on read.

- **HRESP: HRESP Not OK**

Set when the DMA block sees HRESP not OK. Cleared on read.

- **PFNZ: Pause Frame with Non-zero Pause Quantum Received**

Indicates a valid pause has been received that has a non-zero pause quantum field. Cleared on read.

- **PTZ: Pause Time Zero**

Set when either the Pause Time Register at address 0x38 decrements to zero, or when a valid pause frame is received with a zero pause quantum field. Cleared on read.

- **PFTR: Pause Frame Transmitted**

Indicates a pause frame has been successfully transmitted after being initiated from the Network Control Register. Cleared on read.

- **DRQFR: PTP Delay Request Frame Received**

Indicates a PTP delay_req frame has been received. Cleared on read.

- **SFR: PTP Sync Frame Received**

Indicates a PTP sync frame has been received. Cleared on read.

- **DRQFT: PTP Delay Request Frame Transmitted**

Indicates a PTP delay_req frame has been transmitted. Cleared on read.

- **SFT: PTP Sync Frame Transmitted**

Indicates a PTP sync frame has been transmitted. Cleared on read.

- **PDRQFR: PDelay Request Frame Received**

Indicates a PTP pdelay_req frame has been received. Cleared on read.

- **PDRSFR: PDelay Response Frame Received**

Indicates a PTP pdelay_resp frame has been received. Cleared on read.

- **PDRQFT: PDelay Request Frame Transmitted**

Indicates a PTP pdelay_req frame has been transmitted. Cleared on read.

- **PDRSFT: PDelay Response Frame Transmitted**

Indicates a PTP pdelay_resp frame has been transmitted. Cleared on read.

- **SRI: TSU Seconds Register Increment**

Indicates the register has incremented. Cleared on read.

- **WOL: Wake On LAN**

WOL interrupt. Indicates a WOL event has been received.

Downloaded from Arrow.com.

38.8.11 GMAC Interrupt Enable Register

Name: GMAC_IER

Address: 0x40050028

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	WOL	–	SRI	PDRSFT	PDRQFT
23	22	21	20	19	18	17	16
PDRSFR	PDRQFR	SFT	DRQFT	SFR	DRQFR	–	–
15	14	13	12	11	10	9	8
EXINT	PFTR	PTZ	PFNZ	HRESP	ROVR	–	–
7	6	5	4	3	2	1	0
TCOMP	TFC	RLEX	TUR	TXUBR	RXUBR	RCOMP	MFS

This register is write-only and when read will return zero.

The following values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

- **MFS: Management Frame Sent**
- **RCOMP: Receive Complete**
- **RXUBR: RX Used Bit Read**
- **TXUBR: TX Used Bit Read**
- **TUR: Transmit Underrun**
- **RLEX: Retry Limit Exceeded or Late Collision**
- **TFC: Transmit Frame Corruption Due to AHB Error**
- **TCOMP: Transmit Complete**
- **ROVR: Receive Overrun**
- **HRESP: HRESP Not OK**
- **PFNZ: Pause Frame with Non-zero Pause Quantum Received**
- **PTZ: Pause Time Zero**
- **PFTR: Pause Frame Transmitted**
- **EXINT: External Interrupt**
- **DRQFR: PTP Delay Request Frame Received**

Downloaded from Arrow.com.

- **SFR: PTP Sync Frame Received**
- **DRQFT: PTP Delay Request Frame Transmitted**
- **SFT: PTP Sync Frame Transmitted**
- **PDRQFR: PDelay Request Frame Received**
- **PDRSFR: PDelay Response Frame Received**
- **PDRQFT: PDelay Request Frame Transmitted**
- **PDRSFT: PDelay Response Frame Transmitted**
- **SRI: TSU Seconds Register Increment**
- **WOL: Wake On LAN**

Downloaded from Arrow.com.

38.8.12 GMAC Interrupt Disable Register

Name: GMAC_IDR

Address: 0x4005002C

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	WOL	–	SRI	PDRSFT	PDRQFT
23	22	21	20	19	18	17	16
PDRSFR	PDRQFR	SFT	DRQFT	SFR	DRQFR	–	–
15	14	13	12	11	10	9	8
EXINT	PFTR	PTZ	PFNZ	HRESP	ROVR	–	–
7	6	5	4	3	2	1	0
TCOMP	TFC	RLEX	TUR	TXUBR	RXUBR	RCOMP	MFS

This register is write-only and when read will return zero.

The following values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

- **MFS: Management Frame Sent**
- **RCOMP: Receive Complete**
- **RXUBR: RX Used Bit Read**
- **TXUBR: TX Used Bit Read**
- **TUR: Transmit Underrun**
- **RLEX: Retry Limit Exceeded or Late Collision**
- **TFC: Transmit Frame Corruption Due to AHB Error**
- **TCOMP: Transmit Complete**
- **ROVR: Receive Overrun**
- **HRESP: HRESP Not OK**
- **PFNZ: Pause Frame with Non-zero Pause Quantum Received**
- **PTZ: Pause Time Zero**
- **PFTR: Pause Frame Transmitted**
- **EXINT: External Interrupt**
- **DRQFR: PTP Delay Request Frame Received**

Downloaded from Arrow.com.

- **SFR: PTP Sync Frame Received**
- **DRQFT: PTP Delay Request Frame Transmitted**
- **SFT: PTP Sync Frame Transmitted**
- **PDRQFR: PDelay Request Frame Received**
- **PDRSFR: PDelay Response Frame Received**
- **PDRQFT: PDelay Request Frame Transmitted**
- **PDRSFT: PDelay Response Frame Transmitted**
- **SRI: TSU Seconds Register Increment**
- **WOL: Wake On LAN**

Downloaded from Arrow.com.

38.8.13 GMAC Interrupt Mask Register

Name: GMAC_IMR

Address: 0x40050030

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	PDRSFT	PDRQFT
23	22	21	20	19	18	17	16
PDRSFR	PDRQFR	SFT	DRQFT	SFR	DRQFR	–	–
15	14	13	12	11	10	9	8
EXINT	PFTR	PTZ	PFNZ	HRESP	ROVR	–	–
7	6	5	4	3	2	1	0
TCOMP	TFC	RLEX	TUR	TXUBR	RXUBR	RCOMP	MFS

The Interrupt Mask Register is a read-only register indicating which interrupts are masked. All bits are set at reset and can be reset individually by writing to the Interrupt Enable Register or set individually by writing to the Interrupt Disable Register. Having separate address locations for enable and disable saves the need for performing a read modify write when updating the Interrupt Mask Register.

For test purposes there is a write-only function to this register that allows the bits in the Interrupt Status Register to be set or cleared, regardless of the state of the mask register. A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.

The following values are valid for all listed bit names of this register when read:

0: The corresponding interrupt is enabled.

1: The corresponding interrupt is not enabled.

- **MFS: Management Frame Sent**
- **RCOMP: Receive Complete**
- **RXUBR: RX Used Bit Read**
- **TXUBR: TX Used Bit Read**
- **TUR: Transmit Underrun**
- **RLEX: Retry Limit Exceeded**
- **TFC: Transmit Frame Corruption Due to AHB Error**
- **TCOMP: Transmit Complete**
- **ROVR: Receive Overrun**
- **HRESP: HRESP Not OK**
- **PFNZ: Pause Frame with Non-zero Pause Quantum Received**
- **PTZ: Pause Time Zero**

Downloaded from Arrow.com.

- **PFTR: Pause Frame Transmitted**
- **EXINT: External Interrupt**
- **DRQFR: PTP Delay Request Frame Received**
- **SFR: PTP Sync Frame Received**
- **DRQFT: PTP Delay Request Frame Transmitted**
- **SFT: PTP Sync Frame Transmitted**
- **PDRQFR: PDelay Request Frame Received**
- **PDRSFR: PDelay Response Frame Received**
- **PDRQFT: PDelay Request Frame Transmitted**
- **PDRSFT: PDelay Response Frame Transmitted**

Downloaded from Arrow.com.

38.8.14 GMAC PHY Maintenance Register

Name: GMAC_MAN

Address: 0x40050034

Access: Read/Write

31	30	29	28	27	26	25	24
WZO	CLTTO	OP		PHYA			
23	22	21	20	19	18	17	16
PHYA	REGA					WTN	
15	14	13	12	11	10	9	8
DATA							
7	6	5	4	3	2	1	0
DATA							

The PHY Maintenance Register is implemented as a shift register. Writing to the register starts a shift operation which is signalled as complete when bit 2 is set in the Network Status Register. It takes about 2000 MCK cycles to complete, when MDC is set for MCK divide by 32 in the Network Configuration Register. An interrupt is generated upon completion.

During this time, the MSB of the register is output on the MDIO pin and the LSB updated from the MDIO pin with each MDC cycle. This causes transmission of a PHY management frame on MDIO. See Section 22.2.4.5 of the IEEE 802.3 standard.

Reading during the shift operation returns the current contents of the shift register. At the end of management operation, the bits will have shifted back to their original locations. For a read operation, the data bits are updated with data read from the PHY. It is important to write the correct values to the register to ensure a valid PHY management frame is produced.

The MDIO interface can read IEEE 802.3 clause 45 PHYs as well as clause 22 PHYs. To read clause 45 PHYs, bit 30 should be written with a 0 rather than a 1. To write clause 45 PHYs, bits 31:28 should be written as 0x0001. See Table 38-18.

Table 38-18. Clause 22/Clause 45 PHYs Read/Write Access Configuration (GMAC_MAN Bits 31:28)

PHY	Access	Bit Value			
		WZO	CLTTO	OP[1]	OP[0]
Clause 22	Read	0	1	1	0
	Write	0	1	0	1
Clause 45	Read	0	0	1	1
	Write	0	0	0	1
	Read + Address	0	0	1	0

For a description of MDC generation, see Section 38.8.2 "GMAC Network Configuration Register".

- **DATA: PHY Data**

For a write operation this field is written with the data to be written to the PHY. After a read operation this field contains the data read from the PHY.

- **WTN: Write Ten**

Must be written to 10.

- **REGA: Register Address**

Downloaded from Arrow.com.

Specifies the register in the PHY to access.

- **PHYA: PHY Address**

- **OP: Operation**

01: Write

10: Read

- **CLTTO: Clause 22 Operation**

0: Clause 45 operation

1: Clause 22 operation

- **WZO: Write ZERO**

Must be written with 0.

Downloaded from Arrow.com.

38.8.15 GMAC Receive Pause Quantum Register

Name: GMAC_RPQ

Address: 0x40050038

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
RPQ							

7	6	5	4	3	2	1	0
RPQ							

- **RPQ: Received Pause Quantum**

Stores the current value of the Receive Pause Quantum Register which is decremented every 512 bit times.

Downloaded from Arrow.com.

38.8.16 GMAC Transmit Pause Quantum Register

Name: GMAC_TPQ

Address: 0x4005003C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
TPQ							

7	6	5	4	3	2	1	0
TPQ							

- **TPQ: Transmit Pause Quantum**

Written with the pause quantum value for pause frame transmission.

Downloaded from Arrow.com.

38.8.17 GMAC TX Partial Store and Forward Register

Name: GMAC_TPSF

Address: 0x40050040

Access: Read/Write

31	30	29	28	27	26	25	24
ENTXP	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	TPB1ADR			
7	6	5	4	3	2	1	0
TPB1ADR							

- **TPB1ADR: Transmit Partial Store and Forward Address**

Watermark value. Reset = 1.

- **ENTXP: Enable TX Partial Store and Forward Operation**

Downloaded from Arrow.com.

38.8.18 GMAC RX Partial Store and Forward Register

Name: GMAC_RPSF

Address: 0x40050044

Access: Read/Write

31	30	29	28	27	26	25	24
ENRXP	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RPB1ADR			
7	6	5	4	3	2	1	0
RPB1ADR							

- **RPB1ADR: Receive Partial Store and Forward Address**

Watermark value. Reset = 1.

- **ENRXP: Enable RX Partial Store and Forward Operation**

Downloaded from Arrow.com.

38.8.19 GMAC RX Jumbo Frame Max Length Register

Name: GMAC_RJFML

Address: 0x40050048

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	FML					
7	6	5	4	3	2	1	0
FML							

- **FML: Frame Max Length**

Rx jumbo frame maximum length.

Downloaded from Arrow.com.

38.8.20 GMAC Hash Register Bottom

Name: GMAC_HRB

Address: 0x40050080

Access: Read-only

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

The unicast hash enable (UNIHEN) and the multicast hash enable (MITIHEN) bits in the Network Configuration Register (Section 38.8.2 ”GMAC Network Configuration Register”) enable the reception of hash matched frames. See Section 38.6.9 ”Hash Addressing”.

- **ADDR: Hash Address**

The first 32 bits of the Hash Address Register.

Downloaded from Arrow.com.

38.8.21 GMAC Hash Register Top

Name: GMAC_HRT

Address: 0x40050084

Access: Read-only

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

The unicast hash enable (UNIHEN) and the multicast hash enable (MITIHEN) bits in the GMAC Network Configuration Register enable the reception of hash matched frames. See Section 38.6.9 ”Hash Addressing”.

- **ADDR: Hash Address**

Bits 63 to 32 of the Hash Address Register.

Downloaded from Arrow.com.

38.8.22 GMAC Specific Address 1 Bottom Register

Name: GMAC_SAB1

Address: 0x40050088

Access: Read/Write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

The addresses stored in the Specific Address Registers are deactivated at reset or when their corresponding Specific Address Register Bottom is written. They are activated when Specific Address Register Top is written.

- **ADDR: Specific Address 1**

Least significant 32 bits of the destination address, that is, bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

Downloaded from Arrow.com.

38.8.23 GMAC Specific Address 1 Top Register

Name: GMAC_SAT1

Address: 0x4005008C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

The addresses stored in the Specific Address Registers are deactivated at reset or when their corresponding Specific Address Register Bottom is written. They are activated when Specific Address Register Top is written.

- **ADDR: Specific Address 1**

The most significant bits of the destination address, that is, bits 47:32.

Downloaded from Arrow.com.

38.8.24 GMAC Specific Address 2 Bottom Register

Name: GMAC_SAB2

Address: 0x40050090

Access: Read/Write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

The addresses stored in the Specific Address Registers are deactivated at reset or when their corresponding Specific Address Register Bottom is written. They are activated when Specific Address Register Top is written.

- **ADDR: Specific Address 2**

Least significant 32 bits of the destination address, that is, bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

Downloaded from Arrow.com.

38.8.25 GMAC Specific Address 2 Top Register

Name: GMAC_SAT2

Address: 0x40050094

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

The addresses stored in the Specific Address Registers are deactivated at reset or when their corresponding Specific Address Register Bottom is written. They are activated when Specific Address Register Top is written.

- **ADDR: Specific Address 2**

The most significant bits of the destination address, that is, bits 47:32.

Downloaded from Arrow.com.

38.8.26 GMAC Specific Address 3 Bottom Register

Name: GMAC_SAB3

Address: 0x40050098

Access: Read/Write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

The addresses stored in the Specific Address Registers are deactivated at reset or when their corresponding Specific Address Register Bottom is written. They are activated when Specific Address Register Top is written.

- **ADDR: Specific Address 3**

Least significant 32 bits of the destination address, that is, bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

Downloaded from Arrow.com.

38.8.27 GMAC Specific Address 3 Top Register

Name: GMAC_SAT3

Address: 0x4005009C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

The addresses stored in the Specific Address Registers are deactivated at reset or when their corresponding Specific Address Register Bottom is written. They are activated when Specific Address Register Top is written.

- **ADDR: Specific Address 3**

The most significant bits of the destination address, that is, bits 47:32.

Downloaded from Arrow.com.

38.8.28 GMAC Specific Address 4 Bottom Register

Name: GMAC_SAB4

Address: 0x400500A0

Access: Read/Write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

The addresses stored in the Specific Address Registers are deactivated at reset or when their corresponding Specific Address Register Bottom is written. They are activated when Specific Address Register Top is written.

- **ADDR: Specific Address 4**

Least significant 32 bits of the destination address, that is, bits 31:0. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

Downloaded from Arrow.com.

38.8.29 GMAC Specific Address 4 Top Register

Name: GMAC_SAT4

Address: 0x400500A4

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

The addresses stored in the Specific Address Registers are deactivated at reset or when their corresponding Specific Address Register Bottom is written. They are activated when Specific Address Register Top is written.

- **ADDR: Specific Address 4**

The most significant bits of the destination address, that is, bits 47:32.

Downloaded from Arrow.com.

38.8.30 GMAC Type ID Match 1 Register

Name: GMAC_TIDM1

Address: 0x400500A8

Access: Read/Write

31	30	29	28	27	26	25	24
ENID1	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
TID							

7	6	5	4	3	2	1	0
TID							

- **TID: Type ID Match 1**

For use in comparisons with received frames type ID/length frames.

- **ENID1: Enable Copying of TID Matched Frames**

0: TID is not part of the comparison match.

1: TID is processed for the comparison match.

Downloaded from Arrow.com.

38.8.31 GMAC Type ID Match 2 Register

Name: GMAC_TIDM2

Address: 0x400500AC

Access: Read/Write

31	30	29	28	27	26	25	24
ENID2	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
TID							
7	6	5	4	3	2	1	0
TID							

- **TID: Type ID Match 2**

For use in comparisons with received frames type ID/length frames.

- **ENID2: Enable Copying of TID Matched Frames**

0: TID is not part of the comparison match.

1: TID is processed for the comparison match.

Downloaded from Arrow.com.

38.8.32 GMAC Type ID Match 3 Register

Name: GMAC_TIDM3

Address: 0x400500B0

Access: Read/Write

31	30	29	28	27	26	25	24
ENID3	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
TID							
7	6	5	4	3	2	1	0
TID							

- **TID: Type ID Match 3**

For use in comparisons with received frames type ID/length frames.

- **ENID3: Enable Copying of TID Matched Frames**

0: TID is not part of the comparison match.

1: TID is processed for the comparison match.

Downloaded from Arrow.com.

38.8.33 GMAC Type ID Match 4 Register

Name: GMAC_TIDM4

Address: 0x400500B4

Access: Read/Write

31	30	29	28	27	26	25	24
ENID4	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
TID							

7	6	5	4	3	2	1	0
TID							

- **TID: Type ID Match 4**

For use in comparisons with received frames type ID/length frames.

- **ENID4: Enable Copying of TID Matched Frames**

0: TID is not part of the comparison match.

1: TID is processed for the comparison match.

Downloaded from Arrow.com.

38.8.34 GMAC Wake on LAN Register

Name: GMAC_WOL

Address: 0x400500B8

Access: Read/Write

31	30	29	28	27	26	25	24
–							

23	22	21	20	19	18	17	16
–				MTI	SA1	ARP	MAG

15	14	13	12	11	10	9	8
IP							

7	6	5	4	3	2	1	0
IP							

- **IP: ARP Request IP Address**

Wake on LAN ARP request IP address. Written to define the least significant 16 bits of the target IP address that is matched to generate a Wake on LAN event. A value of zero will not generate an event, even if this is matched by the received frame.

- **MAG: Magic Packet Event Enable**

Wake on LAN magic packet event enable.

- **ARP: ARP Request Event Enable**

Wake on LAN ARP request event enable.

- **SA1: Specific Address Register 1 Event Enable**

Wake on LAN Specific Address Register 1 event enable.

- **MTI: Multicast Hash Event Enable**

Wake on LAN multicast hash event enable.

Downloaded from Arrow.com.

38.8.35 GMAC IPG Stretch Register

Name: GMAC_IPGS

Address: 0x400500BC

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
FL							
7	6	5	4	3	2	1	0
FL							

- **FL: Frame Length**

Bits 7:0 are multiplied with the previously transmitted frame length (including preamble). Bits 15:8 +1 divide the frame length. If the resulting number is greater than 96 and bit 28 is set in the Network Configuration Register then the resulting number is used for the transmit inter-packet-gap. 1 is added to bits 15:8 to prevent a divide by zero. See Section 38.6.4 ”MAC Transmit Block”.

Downloaded from Arrow.com.

38.8.36 GMAC Stacked VLAN Register

Name: GMAC_SVLAN

Address: 0x400500C0

Access: Read/Write

31	30	29	28	27	26	25	24
ESVLAN	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
VLAN_TYPE							

7	6	5	4	3	2	1	0
VLAN_TYPE							

- **VLAN_TYPE: User Defined VLAN_TYPE Field**

User defined VLAN_TYPE field. When Stacked VLAN is enabled, the first VLAN tag in a received frame will only be accepted if the VLAN type field is equal to this user defined VLAN_TYPE, OR equal to the standard VLAN type (0x8100). Note that the second VLAN tag of a Stacked VLAN packet will only be matched correctly if its VLAN_TYPE field equals 0x8100.

- **ESVLAN: Enable Stacked VLAN Processing Mode**

0: Disable the stacked VLAN processing mode

1: Enable the stacked VLAN processing mode

Downloaded from Arrow.com.

38.8.37 GMAC Transmit PFC Pause Register

Name: GMAC_TPFCP

Address: 0x400500C4

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
PQ							

7	6	5	4	3	2	1	0
PEV							

- **PEV: Priority Enable Vector**

If bit 17 of the Network Control Register is written with a one then the priority enable vector of the PFC priority based pause frame will be set equal to the value stored in this register [7:0].

- **PQ: Pause Quantum**

If bit 17 of the Network Control Register is written with a one then for each entry equal to zero in the Transmit PFC Pause Register[15:8], the PFC pause frame's pause quantum field associated with that entry will be taken from the Transmit Pause Quantum Register. For each entry equal to one in the Transmit PFC Pause Register [15:8], the pause quantum associated with that entry will be zero.

Downloaded from Arrow.com.

38.8.38 GMAC Specific Address 1 Mask Bottom Register

Name: GMAC_SAMB1

Address: 0x400500C8

Access: Read/Write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

- **ADDR: Specific Address 1 Mask**

Setting a bit to one masks the corresponding bit in the Specific Address 1 Register.

Downloaded from Arrow.com.

38.8.39 GMAC Specific Address Mask 1 Top Register

Name: GMAC_SAMT1

Address: 0x400500CC

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

- **ADDR: Specific Address 1 Mask**

Setting a bit to one masks the corresponding bit in the Specific Address 1 Register.

Downloaded from Arrow.com.

38.8.40 GMAC 1588 Timer Nanosecond Comparison Register

Name: GMAC_NSC

Address: 0x400500DC

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	NANOSEC					

15	14	13	12	11	10	9	8
NANOSEC							

7	6	5	4	3	2	1	0
NANOSEC							

- **NANOSEC: 1588 Timer Nanosecond Comparison Value**

Value is compared to the bits [45:24] of the TSU timer count value (upper 22 bits of nanosecond value).

Downloaded from Arrow.com.

38.8.41 GMAC 1588 Timer Second Comparison Low Register

Name: GMAC_SCL

Address: 0x400500E0

Access: Read/Write

31	30	29	28	27	26	25	24
SEC							

23	22	21	20	19	18	17	16
SEC							

15	14	13	12	11	10	9	8
SEC							

7	6	5	4	3	2	1	0
SEC							

- **SEC: 1588 Timer Second Comparison Value**

Value is compared to seconds value bits [31:0] of the TSU timer count value.

Downloaded from Arrow.com.

38.8.42 GMAC 1588 Timer Second Comparison High Register

Name: GMAC_SCH

Address: 0x400500E4

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
SEC							
7	6	5	4	3	2	1	0
SEC							

- **SEC: 1588 Timer Second Comparison Value**

Value is compared to the top 16 bits (most significant 16 bits [47:32] of seconds value) of the TSU timer count value.

Downloaded from Arrow.com.

38.8.43 GMAC PTP Event Frame Transmitted Seconds High Register

Name: GMAC_EFTSH

Address: 0x400500E8

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
RUD							

7	6	5	4	3	2	1	0
RUD							

- **RUD: Register Update**

The register is updated with the value that the 1588 timer seconds register held when the SFD of a PTP transmit primary event crosses the MII interface. An interrupt is issued when the register is updated.

Downloaded from Arrow.com.

38.8.44 GMAC PTP Event Frame Received Seconds High Register

Name: GMAC_EFRSH

Address: 0x400500EC

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
RUD							
7	6	5	4	3	2	1	0
RUD							

- **RUD: Register Update**

The register is updated with the value that the 1588 timer seconds register held when the SFD of a PTP transmit primary event crosses the MII interface. An interrupt is issued when the register is updated.

Downloaded from Arrow.com.

38.8.45 GMAC PTP Peer Event Frame Transmitted Seconds High Register

Name: GMAC_PEFTSH

Address: 0x400500F0

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
RUD							

7	6	5	4	3	2	1	0
RUD							

- **RUD: Register Update**

The register is updated with the value that the 1588 timer seconds register held when the SFD of a PTP transmit peer event crosses the MII interface. An interrupt is issued when the register is updated.

Downloaded from Arrow.com.

38.8.46 GMAC PTP Peer Event Frame Received Seconds High Register

Name: GMAC_PEFRSH

Address: 0x400500F4

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
RUD							

7	6	5	4	3	2	1	0
RUD							

- **RUD: Register Update**

The register is updated with the value that the 1588 timer seconds register held when the SFD of a PTP transmit peer event crosses the MII interface. An interrupt is issued when the register is updated.

Downloaded from Arrow.com.

38.8.47 GMAC Octets Transmitted Low Register

Name: GMAC_OTLO

Address: 0x40050100

Access: Read-only

31	30	29	28	27	26	25	24
TXO							

23	22	21	20	19	18	17	16
TXO							

15	14	13	12	11	10	9	8
TXO							

7	6	5	4	3	2	1	0
TXO							

When reading the Octets Transmitted and Octets Received Registers, bits 31:0 should be read prior to bits 47:32 to ensure reliable operation.

- **TXO: Transmitted Octets**

Transmitted octets in frame without errors [31:0]. The number of octets transmitted in valid frames of any type. This counter is 48-bits, and is read through two registers. This count does not include octets from automatically generated pause frames.

Downloaded from Arrow.com.

38.8.48 GMAC Octets Transmitted High Register

Name: GMAC_OTHI

Address: 0x40050104

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
TXO							

7	6	5	4	3	2	1	0
TXO							

When reading the Octets Transmitted and Octets Received Registers, bits 31:0 should be read prior to bits 47:32 to ensure reliable operation.

- **TXO: Transmitted Octets**

Transmitted octets in frame without errors [47:32]. The number of octets transmitted in valid frames of any type. This counter is 48-bits, and is read through two registers. This count does not include octets from automatically generated pause frames.

Downloaded from Arrow.com.

38.8.49 GMAC Frames Transmitted Register

Name: GMAC_FT

Address: 0x40050108

Access: Read-only

31	30	29	28	27	26	25	24
FTX							

23	22	21	20	19	18	17	16
FTX							

15	14	13	12	11	10	9	8
FTX							

7	6	5	4	3	2	1	0
FTX							

- **FTX: Frames Transmitted without Error**

Frames transmitted without error. This register counts the number of frames successfully transmitted, i.e., no underrun and not too many retries. Excludes pause frames.

Downloaded from Arrow.com.

38.8.50 GMAC Broadcast Frames Transmitted Register

Name: GMAC_BCFT

Address: 0x4005010C

Access: Read-only

31	30	29	28	27	26	25	24
BFTX							

23	22	21	20	19	18	17	16
BFTX							

15	14	13	12	11	10	9	8
BFTX							

7	6	5	4	3	2	1	0
BFTX							

- **BFTX: Broadcast Frames Transmitted without Error**

Broadcast frames transmitted without error. This register counts the number of broadcast frames successfully transmitted without error, i.e., no underrun and not too many retries. Excludes pause frames.

Downloaded from Arrow.com.

38.8.51 GMAC Multicast Frames Transmitted Register

Name: GMAC_MFT

Address: 0x40050110

Access: Read-only

31	30	29	28	27	26	25	24
MFTX							

23	22	21	20	19	18	17	16
MFTX							

15	14	13	12	11	10	9	8
MFTX							

7	6	5	4	3	2	1	0
MFTX							

- **MFTX: Multicast Frames Transmitted without Error**

This register counts the number of multicast frames successfully transmitted without error, i.e., no underrun and not too many retries. Excludes pause frames.

Downloaded from Arrow.com.

38.8.52 GMAC Pause Frames Transmitted Register

Name: GMAC_PFT

Address: 0x40050114

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
PFTX							

7	6	5	4	3	2	1	0
PFTX							

- **PFTX: Pause Frames Transmitted Register**

This register counts the number of pause frames transmitted. Only pause frames triggered by the register interface or through the external pause pins are counted as pause frames. Pause frames received through the FIFO interface are counted in the frames transmitted counter.

Downloaded from Arrow.com.

38.8.53 GMAC 64 Byte Frames Transmitted Register

Name: GMAC_BFT64

Address: 0x40050118

Access: Read-only

31	30	29	28	27	26	25	24
NFTX							

23	22	21	20	19	18	17	16
NFTX							

15	14	13	12	11	10	9	8
NFTX							

7	6	5	4	3	2	1	0
NFTX							

- **NFTX: 64 Byte Frames Transmitted without Error**

This register counts the number of 64 byte frames successfully transmitted without error, i.e., no underrun and not too many retries. Excludes pause frames.

Downloaded from Arrow.com.

38.8.54 GMAC 65 to 127 Byte Frames Transmitted Register

Name: GMAC_TBFT127

Address: 0x4005011C

Access: Read-only

31	30	29	28	27	26	25	24
NFTX							

23	22	21	20	19	18	17	16
NFTX							

15	14	13	12	11	10	9	8
NFTX							

7	6	5	4	3	2	1	0
NFTX							

- **NFTX: 65 to 127 Byte Frames Transmitted without Error**

This register counts the number of 65 to 127 byte frames successfully transmitted without error, i.e., no underrun and not too many retries. Excludes pause frames.

Downloaded from Arrow.com.

38.8.55 GMAC 128 to 255 Byte Frames Transmitted Register

Name: GMAC_TBFT255

Address: 0x40050120

Access: Read-only

31	30	29	28	27	26	25	24
NFTX							

23	22	21	20	19	18	17	16
NFTX							

15	14	13	12	11	10	9	8
NFTX							

7	6	5	4	3	2	1	0
NFTX							

- **NFTX: 128 to 255 Byte Frames Transmitted without Error**

This register counts the number of 128 to 255 byte frames successfully transmitted without error, i.e., no underrun and not too many retries.

Downloaded from Arrow.com.

38.8.56 GMAC 256 to 511 Byte Frames Transmitted Register

Name: GMAC_TBFT511

Address: 0x40050124

Access: Read-only

31	30	29	28	27	26	25	24
NFTX							

23	22	21	20	19	18	17	16
NFTX							

15	14	13	12	11	10	9	8
NFTX							

7	6	5	4	3	2	1	0
NFTX							

- **NFTX: 256 to 511 Byte Frames Transmitted without Error**

This register counts the number of 256 to 511 byte frames successfully transmitted without error, i.e., no underrun and not too many retries.

Downloaded from Arrow.com.

38.8.57 GMAC 512 to 1023 Byte Frames Transmitted Register

Name: GMAC_TBFT1023

Address: 0x40050128

Access: Read-only

31	30	29	28	27	26	25	24
NFTX							

23	22	21	20	19	18	17	16
NFTX							

15	14	13	12	11	10	9	8
NFTX							

7	6	5	4	3	2	1	0
NFTX							

- **NFTX: 512 to 1023 Byte Frames Transmitted without Error**

This register counts the number of 512 to 1023 byte frames successfully transmitted without error, i.e., no underrun and not too many retries.

Downloaded from Arrow.com.

38.8.58 GMAC 1024 to 1518 Byte Frames Transmitted Register

Name: GMAC_TBFT1518

Address: 0x4005012C

Access: Read-only

31	30	29	28	27	26	25	24
NFTX							

23	22	21	20	19	18	17	16
NFTX							

15	14	13	12	11	10	9	8
NFTX							

7	6	5	4	3	2	1	0
NFTX							

- **NFTX: 1024 to 1518 Byte Frames Transmitted without Error**

This register counts the number of 1024 to 1518 byte frames successfully transmitted without error, i.e., no underrun and not too many retries.

Downloaded from Arrow.com.

38.8.59 GMAC Greater Than 1518 Byte Frames Transmitted Register

Name: GMAC_GTBFT1518

Address: 0x40050130

Access: Read-only

31	30	29	28	27	26	25	24
NFTX							

23	22	21	20	19	18	17	16
NFTX							

15	14	13	12	11	10	9	8
NFTX							

7	6	5	4	3	2	1	0
NFTX							

- **NFTX: Greater than 1518 Byte Frames Transmitted without Error**

This register counts the number of 1518 or above byte frames successfully transmitted without error i.e., no underrun and not too many retries.

Downloaded from Arrow.com.

38.8.60 GMAC Transmit Underruns Register

Name: GMAC_TUR

Address: 0x40050134

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXUNR	

7	6	5	4	3	2	1	0
TXUNR							

- **TXUNR: Transmit Underruns**

This register counts the number of frames not transmitted due to a transmit underrun. If this register is incremented then no other statistics register is incremented.

Downloaded from Arrow.com.

38.8.61 GMAC Single Collision Frames Register

Name: GMAC_SCF

Address: 0x40050138

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	SCOL	

15	14	13	12	11	10	9	8
SCOL							

7	6	5	4	3	2	1	0
SCOL							

- **SCOL: Single Collision**

This register counts the number of frames experiencing a single collision before being successfully transmitted i.e., no underrun.

Downloaded from Arrow.com.

38.8.62 GMAC Multiple Collision Frames Register

Name: GMAC_MCF

Address: 0x4005013C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	MCOL	

15	14	13	12	11	10	9	8
MCOL							

7	6	5	4	3	2	1	0
MCOL							

- **MCOL: Multiple Collision**

This register counts the number of frames experiencing between two and fifteen collisions prior to being successfully transmitted, i.e., no underrun and not too many retries.

Downloaded from Arrow.com.

38.8.63 GMAC Excessive Collisions Register

Name: GMAC_EC

Address: 0x40050140

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	XCOL	

7	6	5	4	3	2	1	0
XCOL							

- **XCOL: Excessive Collisions**

This register counts the number of frames that failed to be transmitted because they experienced 16 collisions.

Downloaded from Arrow.com.

38.8.64 GMAC Late Collisions Register

Name: GMAC_LC

Address: 0x40050144

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	LCOL	
7	6	5	4	3	2	1	0
LCOL							

- **LCOL: Late Collisions**

This register counts the number of late collisions occurring after the slot time (512 bits) has expired. In 10/100 mode, late collisions are counted twice i.e., both as a collision and a late collision.

Downloaded from Arrow.com.

38.8.65 GMAC Deferred Transmission Frames Register

Name: GMAC_DTF

Address: 0x40050148

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	DEFT	

15	14	13	12	11	10	9	8
DEFT							

7	6	5	4	3	2	1	0
DEFT							

- **DEFT: Deferred Transmission**

This register counts the number of frames experiencing deferral due to carrier sense being active on their first attempt at transmission. Frames involved in any collision are not counted nor are frames that experienced a transmit underrun.

Downloaded from Arrow.com.

38.8.66 GMAC Carrier Sense Errors Register

Name: GMAC_CSE

Address: 0x4005014C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	CSR	

7	6	5	4	3	2	1	0
CSR							

- **CSR: Carrier Sense Error**

This register counts the number of frames transmitted where carrier sense was not seen during transmission or where carrier sense was deasserted after being asserted in a transmit frame without collision (no underrun). Only incremented in half duplex mode. The only effect of a carrier sense error is to increment this register. The behavior of the other statistics registers is unaffected by the detection of a carrier sense error.

Downloaded from Arrow.com.

38.8.67 GMAC Octets Received Low Register

Name: GMAC_ORLO

Address: 0x40050150

Access: Read-only

31	30	29	28	27	26	25	24
RXO							

23	22	21	20	19	18	17	16
RXO							

15	14	13	12	11	10	9	8
RXO							

7	6	5	4	3	2	1	0
RXO							

When reading the Octets Transmitted and Octets Received Registers, bits [31:0] should be read prior to bits [47:32] to ensure reliable operation.

- **RXO: Received Octets**

Received octets in frame without errors [31:0]. The number of octets received in valid frames of any type. This counter is 48-bits and is read through two registers. This count does not include octets from pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

Downloaded from Arrow.com.

38.8.68 GMAC Octets Received High Register

Name: GMAC_ORHI

Address: 0x40050154

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
RXO							

7	6	5	4	3	2	1	0
RXO							

When reading the Octets Transmitted and Octets Received Registers, bits 31:0 should be read prior to bits 47:32 to ensure reliable operation.

- **RXO: Received Octets**

Received octets in frame without errors [47:32]. The number of octets received in valid frames of any type. This counter is 48-bits and is read through two registers. This count does not include octets from pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

Downloaded from Arrow.com.

38.8.69 GMAC Frames Received Register

Name: GMAC_FR

Address: 0x40050158

Access: Read-only

31	30	29	28	27	26	25	24
FRX							

23	22	21	20	19	18	17	16
FRX							

15	14	13	12	11	10	9	8
FRX							

7	6	5	4	3	2	1	0
FRX							

- **FRX: Frames Received without Error**

Frames received without error. This register counts the number of frames successfully received. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

Downloaded from Arrow.com.

38.8.70 GMAC Broadcast Frames Received Register

Name: GMAC_BCFR

Address: 0x4005015C

Access: Read-only

31	30	29	28	27	26	25	24
BFRX							

23	22	21	20	19	18	17	16
BFRX							

15	14	13	12	11	10	9	8
BFRX							

7	6	5	4	3	2	1	0
BFRX							

- **BFRX: Broadcast Frames Received without Error**

Broadcast frames received without error. This register counts the number of broadcast frames successfully received. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

Downloaded from Arrow.com.

38.8.71 GMAC Multicast Frames Received Register

Name: GMAC_MFR

Address: 0x40050160

Access: Read-only

31	30	29	28	27	26	25	24
MFRX							

23	22	21	20	19	18	17	16
MFRX							

15	14	13	12	11	10	9	8
MFRX							

7	6	5	4	3	2	1	0
MFRX							

- **MFRX: Multicast Frames Received without Error**

This register counts the number of multicast frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

Downloaded from Arrow.com.

38.8.72 GMAC Pause Frames Received Register

Name: GMAC_PFR

Address: 0x40050164

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
PFRX							
7	6	5	4	3	2	1	0
PFRX							

- **PFRX: Pause Frames Received Register**

This register counts the number of pause frames received without error.

Downloaded from Arrow.com.

38.8.73 GMAC 64 Byte Frames Received Register

Name: GMAC_BFR64

Address: 0x40050168

Access: Read-only

31	30	29	28	27	26	25	24
NFRX							

23	22	21	20	19	18	17	16
NFRX							

15	14	13	12	11	10	9	8
NFRX							

7	6	5	4	3	2	1	0
NFRX							

- **NFRX: 64 Byte Frames Received without Error**

This register counts the number of 64 byte frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

Downloaded from Arrow.com.

38.8.74 GMAC 65 to 127 Byte Frames Received Register

Name: GMAC_TBFR127

Address: 0x4005016C

Access: Read-only

31	30	29	28	27	26	25	24
NFRX							

23	22	21	20	19	18	17	16
NFRX							

15	14	13	12	11	10	9	8
NFRX							

7	6	5	4	3	2	1	0
NFRX							

- **NFRX: 65 to 127 Byte Frames Received without Error**

This register counts the number of 65 to 127 byte frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

Downloaded from Arrow.com.

38.8.75 GMAC 128 to 255 Byte Frames Received Register

Name: GMAC_TBFR255

Address: 0x40050170

Access: Read-only

31	30	29	28	27	26	25	24
NFRX							

23	22	21	20	19	18	17	16
NFRX							

15	14	13	12	11	10	9	8
NFRX							

7	6	5	4	3	2	1	0
NFRX							

- **NFRX: 128 to 255 Byte Frames Received without Error**

This register counts the number of 128 to 255 byte frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

Downloaded from Arrow.com.

38.8.76 GMAC 256 to 511 Byte Frames Received Register

Name: GMAC_TBFR511

Address: 0x40050174

Access: Read-only

31	30	29	28	27	26	25	24
NFRX							

23	22	21	20	19	18	17	16
NFRX							

15	14	13	12	11	10	9	8
NFRX							

7	6	5	4	3	2	1	0
NFRX							

- **NFRX: 256 to 511 Byte Frames Received without Error**

This register counts the number of 256 to 511 byte frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

Downloaded from Arrow.com.

38.8.77 GMAC 512 to 1023 Byte Frames Received Register

Name: GMAC_TBFR1023

Address: 0x40050178

Access: Read-only

31	30	29	28	27	26	25	24
NFRX							

23	22	21	20	19	18	17	16
NFRX							

15	14	13	12	11	10	9	8
NFRX							

7	6	5	4	3	2	1	0
NFRX							

- **NFRX: 512 to 1023 Byte Frames Received without Error**

This register counts the number of 512 to 1023 byte frames successfully received without error. Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory.

Downloaded from Arrow.com.

38.8.78 GMAC 1024 to 1518 Byte Frames Received Register

Name: GMAC_TBFR1518

Address: 0x4005017C

Access: Read-only

31	30	29	28	27	26	25	24
NFRX							

23	22	21	20	19	18	17	16
NFRX							

15	14	13	12	11	10	9	8
NFRX							

7	6	5	4	3	2	1	0
NFRX							

- **NFRX: 1024 to 1518 Byte Frames Received without Error**

This register counts the number of 1024 to 1518 byte frames successfully received without error, i.e., no underrun and not too many retries.

Downloaded from Arrow.com.

38.8.79 GMAC 1519 to Maximum Byte Frames Received Register

Name: GMAC_TMXBFR

Address: 0x40050180

Access: Read-only

31	30	29	28	27	26	25	24
NFRX							

23	22	21	20	19	18	17	16
NFRX							

15	14	13	12	11	10	9	8
NFRX							

7	6	5	4	3	2	1	0
NFRX							

- **NFRX: 1519 to Maximum Byte Frames Received without Error**

This register counts the number of 1519 byte or above frames successfully received without error. Maximum frame size is determined by the Network Configuration Register bit 8 (1536 maximum frame size) or bit 3 (jumbo frame size). Excludes pause frames, and is only incremented if the frame is successfully filtered and copied to memory. See Section 38.8.2 ”GMAC Network Configuration Register”.

Downloaded from Arrow.com.

38.8.80 GMAC Undersized Frames Received Register

Name: GMAC_UFR

Address: 0x40050184

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	UFRX	

7	6	5	4	3	2	1	0
UFRX							

- **UFRX: Undersize Frames Received**

This register counts the number of frames received less than 64 bytes in length (10/100 mode, full duplex) that do not have either a CRC error or an alignment error.

Downloaded from Arrow.com.

38.8.81 GMAC Oversized Frames Received Register

Name: GMAC_OFR

Address: 0x40050188

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	OFRX	

7	6	5	4	3	2	1	0
OFRX							

- **OFRX: Oversized Frames Received**

This register counts the number of frames received exceeding 1518 bytes (1536 bytes if bit 8 is set in the Network Configuration Register) in length but do not have either a CRC error, an alignment error nor a receive symbol error. See Section 38.8.2 ”GMAC Network Configuration Register”.

Downloaded from Arrow.com.

38.8.82 GMAC Jabbers Received Register

Name: GMAC_JR

Address: 0x4005018C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	JRX	
7	6	5	4	3	2	1	0
JRX							

- **JRX: Jabbers Received**

The register counts the number of frames received exceeding 1518 bytes in length (1536 if bit 8 is set in Network Configuration Register) and have either a CRC error, an alignment error or a receive symbol error. See Section 38.8.2 ”GMAC Network Configuration Register”.

Downloaded from Arrow.com.

38.8.83 GMAC Frame Check Sequence Errors Register

Name: GMAC_FCSE

Address: 0x40050190

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	FCKR	

7	6	5	4	3	2	1	0
FCKR							

- **FCKR: Frame Check Sequence Errors**

The register counts frames that are an integral number of bytes, have bad CRC and are between 64 and 1518 bytes in length (1536 if bit 8 is set in Network Configuration Register). This register is also incremented if a symbol error is detected and the frame is of valid length and has an integral number of bytes.

This register is incremented for a frame with bad FCS, regardless of whether it is copied to memory due to ignore FCS mode being enabled in bit 26 of the Network Configuration Register. See Section 38.8.2 ”GMAC Network Configuration Register”.

Downloaded from Arrow.com.

38.8.84 GMAC Length Field Frame Errors Register

Name: GMAC_LFFE

Address: 0x40050194

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	LFER	

7	6	5	4	3	2	1	0
LFER							

- **LFER: Length Field Frame Errors**

This register counts the number of frames received that have a measured length shorter than that extracted from the length field (bytes 13 and 14). This condition is only counted if the value of the length field is less than 0x0600, the frame is not of excessive length and checking is enabled through bit 16 of the Network Configuration Register. See Section 38.8.2 ”GMAC Network Configuration Register”.

Downloaded from Arrow.com.

38.8.85 GMAC Receive Symbol Errors Register

Name: GMAC_RSE

Address: 0x40050198

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	RXSE	

7	6	5	4	3	2	1	0
RXSE							

- **RXSE: Receive Symbol Errors**

This register counts the number of frames that had GRXER asserted during reception. For 10/100 mode symbol errors are counted regardless of frame length checks. Receive symbol errors will also be counted as an FCS or alignment error if the frame is between 64 and 1518 bytes (1536 bytes if bit 8 is set in the Network Configuration Register). If the frame is larger it will be recorded as a jabber error. See Section 38.8.2 ”GMAC Network Configuration Register”.

Downloaded from Arrow.com.

38.8.86 GMAC Alignment Errors Register

Name: GMAC_AE

Address: 0x4005019C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	AER	

7	6	5	4	3	2	1	0
AER							

- **AER: Alignment Errors**

This register counts the frames that are not an integral number of bytes long and have bad CRC when their length is truncated to an integral number of bytes and are between 64 and 1518 bytes in length (1536 if bit 8 is set in Network Configuration Register). This register is also incremented if a symbol error is detected and the frame is of valid length and does not have an integral number of bytes. See Section 38.8.2 ”GMAC Network Configuration Register”.

Downloaded from Arrow.com.

38.8.87 GMAC Receive Resource Errors Register

Name: GMAC_RRE

Address: 0x400501A0

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	RXRER	

15	14	13	12	11	10	9	8
RXRER							

7	6	5	4	3	2	1	0
RXRER							

- **RXRER: Receive Resource Errors**

This register counts frames that are not an integral number of bytes long and have bad CRC when their length is truncated to an integral number of bytes and are between 64 and 1518 bytes in length (1536 if bit 8 is set in Network Configuration Register). This register is also incremented if a symbol error is detected and the frame is of valid length and does not have an integral number of bytes. See Section 38.8.2 ”GMAC Network Configuration Register”.

Downloaded from Arrow.com.

38.8.88 GMAC Receive Overruns Register

Name: GMAC_ROE

Address: 0x400501A4

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	RXOVR	
7	6	5	4	3	2	1	0
RXOVR							

- **RXOVR: Receive Overruns**

This register counts the number of frames that are address recognized but were not copied to memory due to a receive overrun.

Downloaded from Arrow.com.

38.8.89 GMAC IP Header Checksum Errors Register

Name: GMAC_IHCE

Address: 0x400501A8

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
HCKER							

- **HCKER: IP Header Checksum Errors**

This register counts the number of frames discarded due to an incorrect IP header checksum, but are between 64 and 1518 bytes (1536 bytes if bit 8 is set in the Network Configuration Register) and do not have a CRC error, an alignment error, nor a symbol error.

Downloaded from Arrow.com.

38.8.90 GMAC TCP Checksum Errors Register

Name: GMAC_TCE

Address: 0x400501AC

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
TCKER							

- **TCKER: TCP Checksum Errors**

This register counts the number of frames discarded due to an incorrect TCP checksum, but are between 64 and 1518 bytes (1536 bytes if bit 8 is set in the Network Configuration Register) and do not have a CRC error, an alignment error, nor a symbol error.

Downloaded from Arrow.com.

38.8.91 GMAC UDP Checksum Errors Register

Name: GMAC_UCE

Address: 0x400501B0

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
UCKER							

- **UCKER: UDP Checksum Errors**

This register counts the number of frames discarded due to an incorrect UDP checksum, but are between 64 and 1518 bytes (1536 bytes if bit 8 is set in the Network Configuration Register) and do not have a CRC error, an alignment error, nor a symbol error.

Downloaded from Arrow.com.

38.8.92 GMAC 1588 Timer Increment Sub-nanoseconds Register

Name: GMAC_TISUBN

Address: 0x400501BC

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
LSBTIR							
7	6	5	4	3	2	1	0
LSBTIR							

- **LSBTIR: Lower Significant Bits of Timer Increment Register**

Lower significant bits of Timer Increment Register[15:0] giving a 24-bit timer_increment counter. These bits are the sub-ns value which the 1588 timer will be incremented each clock cycle. Bit n = $2^{(n-16)}$ nsec giving a resolution of approximately $15.2E^{-15}$ sec.

Downloaded from Arrow.com.

38.8.93 GMAC 1588 Timer Seconds High Register

Name: GMAC_TSH

Address: 0x400501C0

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
TCS							

7	6	5	4	3	2	1	0
TCS							

- **TCS: Timer Count in Seconds**

This register is writable. It increments by one when the 1588 nanoseconds counter counts to one second. It may also be incremented when the Timer Adjust Register is written.

Downloaded from Arrow.com.

38.8.94 GMAC 1588 Timer Seconds Low Register

Name: GMAC_TSL

Address: 0x400501D0

Access: Read/Write

31	30	29	28	27	26	25	24
TCS							

23	22	21	20	19	18	17	16
TCS							

15	14	13	12	11	10	9	8
TCS							

7	6	5	4	3	2	1	0
TCS							

- **TCS: Timer Count in Seconds**

This register is writable. It increments by one when the 1588 nanoseconds counter counts to one second. It may also be incremented when the Timer Adjust Register is written.

Downloaded from Arrow.com.

38.8.95 GMAC 1588 Timer Nanoseconds Register

Name: GMAC_TN

Address: 0x400501D4

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	TNS					

23	22	21	20	19	18	17	16
TNS							

15	14	13	12	11	10	9	8
TNS							

7	6	5	4	3	2	1	0
TNS							

- **TNS: Timer Count in Nanoseconds**

This register is writable. It can also be adjusted by writes to the 1588 Timer Adjust Register. It increments by the value of the 1588 Timer Increment Register each clock cycle.

Downloaded from Arrow.com.

38.8.96 GMAC 1588 Timer Adjust Register

Name: GMAC_TA

Address: 0x400501D8

Access: Write-only

31	30	29	28	27	26	25	24
ADJ	–	ITDT					

23	22	21	20	19	18	17	16
ITDT							

15	14	13	12	11	10	9	8
ITDT							

7	6	5	4	3	2	1	0
ITDT							

- **ITDT: Increment/Decrement**

The number of nanoseconds to increment or decrement the 1588 Timer Nanoseconds Register. If necessary, the 1588 Seconds Register will be incremented or decremented.

- **ADJ: Adjust 1588 Timer**

Write as one to subtract from the 1588 timer. Write as zero to add to it.

Downloaded from Arrow.com.

38.8.97 GMAC 1588 Timer Increment Register

Name: GMAC_TI

Address: 0x400501DC

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
NIT							

15	14	13	12	11	10	9	8
ACNS							

7	6	5	4	3	2	1	0
CNS							

- **CNS: Count Nanoseconds**

A count of nanoseconds by which the 1588 Timer Nanoseconds Register will be incremented each clock cycle.

- **ACNS: Alternative Count Nanoseconds**

Alternative count of nanoseconds by which the 1588 Timer Nanoseconds Register will be incremented each clock cycle.

- **NIT: Number of Increments**

The number of increments after which the alternative increment is used.

Downloaded from Arrow.com.

38.8.98 GMAC PTP Event Frame Transmitted Seconds Low Register

Name: GMAC_EFTSL

Address: 0x400501E0

Access: Read-only

31	30	29	28	27	26	25	24
RUD							

23	22	21	20	19	18	17	16
RUD							

15	14	13	12	11	10	9	8
RUD							

7	6	5	4	3	2	1	0
RUD							

- **RUD: Register Update**

The register is updated with the value that the 1588 Timer Seconds Register holds when the SFD of a PTP transmit primary event crosses the MII interface. An interrupt is issued when the register is updated.

Downloaded from Arrow.com.

38.8.99 GMAC PTP Event Frame Transmitted Nanoseconds Register

Name: GMAC_EFTN

Address: 0x400501E4

Access: Read-only

31	30	29	28	27	26	25	24
–	–	RUD					

23	22	21	20	19	18	17	16
RUD							

15	14	13	12	11	10	9	8
RUD							

7	6	5	4	3	2	1	0
RUD							

- **RUD: Register Update**

The register is updated with the value that the 1588 Timer Nanoseconds Register holds when the SFD of a PTP transmit primary event crosses the MII interface. An interrupt is issued when the register is updated.

Downloaded from Arrow.com.

38.8.100 GMAC PTP Event Frame Received Seconds Low Register

Name: GMAC_EFRSL

Address: 0x400501E8

Access: Read-only

31	30	29	28	27	26	25	24
RUD							

23	22	21	20	19	18	17	16
RUD							

15	14	13	12	11	10	9	8
RUD							

7	6	5	4	3	2	1	0
RUD							

- **RUD: Register Update**

The register is updated with the value that the 1588 Timer Seconds Register holds when the SFD of a PTP receive primary event crosses the MII interface. An interrupt is issued when the register is updated.

Downloaded from Arrow.com.

38.8.101 GMAC PTP Event Frame Received Nanoseconds Register

Name: GMAC_EFRN

Address: 0x400501EC

Access: Read-only

31	30	29	28	27	26	25	24
–	–	RUD					

23	22	21	20	19	18	17	16
RUD							

15	14	13	12	11	10	9	8
RUD							

7	6	5	4	3	2	1	0
RUD							

- **RUD: Register Update**

The register is updated with the value that the 1588 Timer Nanoseconds Register holds when the SFD of a PTP receive primary event crosses the MII interface. An interrupt is issued when the register is updated.

Downloaded from Arrow.com.

38.8.102 GMAC PTP Peer Event Frame Transmitted Seconds Low Register

Name: GMAC_PEFTSL

Address: 0x400501F0

Access: Read-only

31	30	29	28	27	26	25	24
RUD							

23	22	21	20	19	18	17	16
RUD							

15	14	13	12	11	10	9	8
RUD							

7	6	5	4	3	2	1	0
RUD							

- **RUD: Register Update**

The register is updated with the value that the 1588 Timer Seconds Register holds when the SFD of a PTP transmit peer event crosses the MII interface. An interrupt is issued when the register is updated.

Downloaded from Arrow.com.

38.8.103 GMAC PTP Peer Event Frame Transmitted Nanoseconds Register

Name: GMAC_PEFTN

Address: 0x400501F4

Access: Read-only

31	30	29	28	27	26	25	24
–	–	RUD					

23	22	21	20	19	18	17	16
RUD							

15	14	13	12	11	10	9	8
RUD							

7	6	5	4	3	2	1	0
RUD							

- **RUD: Register Update**

The register is updated with the value that the 1588 Timer Nanoseconds Register holds when the SFD of a PTP transmit peer event crosses the MII interface. An interrupt is issued when the register is updated.

Downloaded from Arrow.com.

38.8.104 GMAC PTP Peer Event Frame Received Seconds Low Register

Name: GMAC_PEFRSL

Address: 0x400501F8

Access: Read-only

31	30	29	28	27	26	25	24
RUD							

23	22	21	20	19	18	17	16
RUD							

15	14	13	12	11	10	9	8
RUD							

7	6	5	4	3	2	1	0
RUD							

- **RUD: Register Update**

The register is updated with the value that the 1588 Timer Seconds Register holds when the SFD of a PTP receive primary event crosses the MII interface. An interrupt is issued when the register is updated.

Downloaded from Arrow.com.

38.8.105 GMAC PTP Peer Event Frame Received Nanoseconds Register

Name: GMAC_PEFRN

Address: 0x400501FC

Access: Read-only

31	30	29	28	27	26	25	24
–	–	RUD					

23	22	21	20	19	18	17	16
RUD							

15	14	13	12	11	10	9	8
RUD							

7	6	5	4	3	2	1	0
RUD							

- **RUD: Register Update**

The register is updated with the value that the 1588 Timer Nanoseconds Register holds when the SFD of a PTP receive primary event crosses the MII interface. An interrupt is issued when the register is updated.

Downloaded from Arrow.com.

38.8.106 GMAC Interrupt Status Register Priority Queue x

Name: GMAC_ISRPQx[x=1..2]

Address: 0x40050400

Access: Read-only

31	30	29	28	27	26	25	24
–							

23	22	21	20	19	18	17	16
–							

15	14	13	12	11	10	9	8
–				HRESP	ROVR	–	–

7	6	5	4	3	2	1	0
TCOMP	TFC	RLEX	–	–	RXUBR	RCOMP	–

- **RCOMP: Receive Complete**
- **RXUBR: RX Used Bit Read**
- **RLEX: Retry Limit Exceeded or Late Collision**
- **TFC: Transmit Frame Corruption Due to AHB Error**

Transmit frame corruption due to AHB error—set if an error occurs whilst midway through reading transmit frame from the AHB, including HRESP errors and buffers exhausted mid frame.

- **TCOMP: Transmit Complete**
- **ROVR: Receive Overrun**
- **HRESP: HRESP Not OK**

Downloaded from Arrow.com.

38.8.107 GMAC Transmit Buffer Queue Base Address Register Priority Queue x

Name: GMAC_TBQBAPQx[x=1..2]

Address: 0x40050440

Access: Read/Write

31	30	29	28	27	26	25	24
TXBQBA							

23	22	21	20	19	18	17	16
TXBQBA							

15	14	13	12	11	10	9	8
TXBQBA							

7	6	5	4	3	2	1	0
TXBQBA						–	–

These registers hold the start address of the transmit buffer queues (transmit buffers descriptor lists) for the additional queues used when priority queues are employed.

- **TXBQBA: Transmit Buffer Queue Base Address**

Written with the address of the start of the transmit queue.

Downloaded from Arrow.com.

38.8.108 GMAC Receive Buffer Queue Base Address Register Priority Queue x

Name: GMAC_RBQBAPQx[x=1..2]

Address: 0x40050480

Access: Read/Write

31	30	29	28	27	26	25	24
RXBQBA							

23	22	21	20	19	18	17	16
RXBQBA							

15	14	13	12	11	10	9	8
RXBQBA							

7	6	5	4	3	2	1	0
RXBQBA						–	–

These registers hold the start address of the receive buffer queues (receive buffers descriptor lists) for the additional queues used when priority queues are employed.

- **RXBQBA: Receive Buffer Queue Base Address**

Written with the address of the start of the receive queue.

Downloaded from Arrow.com.

38.8.109 GMAC Receive Buffer Size Register Priority Queue x

Name: GMAC_RBSRPQx[x=1..2]

Address: 0x400504A0

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
RBS							
7	6	5	4	3	2	1	0
RBS							

- **RBS: Receive Buffer Size**

DMA receive buffer size in AHB system memory. The value defined by these bits determines the size of buffer to use in main AHB system memory when writing received data.

The value is defined in multiples of 64 bytes such that a value of 0x01 corresponds to buffers of 64 bytes, 0x02 corresponds to 128 bytes etc.

For example:

0x02: 128 bytes

0x18: 1536 bytes (1 × max length frame/buffer)

0xA0: 10240 bytes (1 × 10K jumbo frame/buffer)

Note that this value should never be written as zero.

Downloaded from Arrow.com.

38.8.110 GMAC Credit-Based Shaping Control Register

Name: GMAC_CBSCR

Address: 0x400504BC

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	QAE	QBE

- **• QAE: Queue A CBS Enable**

0: Credit-based shaping on the second highest priority queue (queue A) is disabled.

1: Credit-based shaping on the second highest priority queue (queue A) is enabled.

- **• QBE: Queue B CBS Enable**

0: Credit-based shaping on the highest priority queue (queue B) is disabled.

1: Credit-based shaping on the highest priority queue (queue B) is enabled.

Downloaded from Arrow.com.

38.8.111 GMAC Credit-Based Shaping IdleSlope Register for Queue A

Name: GMAC_CBSISQA

Address: 0x400504C0

Access: Read/Write

31	30	29	28	27	26	25	24
IS							

23	22	21	20	19	18	17	16
IS							

15	14	13	12	11	10	9	8
IS							

7	6	5	4	3	2	1	0
IS							

Credit-based shaping must be disabled in GMAC_CBSCR before updating this register.

- **IS: IdleSlope**

IdleSlope value for queue A in bytes/second.

The IdleSlope value is defined as the rate of change of credit when a packet is waiting to be sent. This must not exceed the port transmit rate which is dependent on the speed of operation, e.g., 1Gb/second portTransmitRate = 32'h07735940

If 50% of bandwidth was to be allocated to a particular queue in 1Gb/second mode, then the IdleSlope value for that queue would be calculated as 32'h07735940 / 2.

Downloaded from Arrow.com.

38.8.112 GMAC Credit-Based Shaping IdleSlope Register for Queue B

Name: GMAC_CBSISQB

Address: 0x400504C4

Access: Read/Write

31	30	29	28	27	26	25	24
IS							

23	22	21	20	19	18	17	16
IS							

15	14	13	12	11	10	9	8
IS							

7	6	5	4	3	2	1	0
IS							

Credit-based shaping must be disabled in GMAC_CBSCR before updating this register.

- **IS: IdleSlope**

IdleSlope value for queue B in bytes/second.

The IdleSlope value is defined as the rate of change of credit when a packet is waiting to be sent. This must not exceed the port transmit rate which is dependent on the speed of operation, e.g., 1Gb/second portTransmitRate = 32'h07735940

If 50% of bandwidth was to be allocated to a particular queue in 1Gb/sec mode, then the IdleSlope value for that queue would be calculated as 32'h07735940 / 2

Downloaded from Arrow.com.

38.8.113 GMAC Screening Type 1 Register x Priority Queue

Name: GMAC_ST1RPQx[x=0..3]

Address: 0x40050500

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	UDPE	DSTCE	UDPM			

23	22	21	20	19	18	17	16
UDPM							

15	14	13	12	11	10	9	8
UDPM				DSTCM			

7	6	5	4	3	2	1	0
DSTCM				–	QNB		

Screening type 1 registers are used to allocate up to 3 priority queues to received frames based on certain IP or UDP fields of incoming frames.

- **QNB: Queue Number (0–2)**

If a match is successful, then the queue value programmed in bits 2:0 is allocated to the frame.

- **DSTCM: Differentiated Services or Traffic Class Match**

When DS/TC match enable is set (bit 28), the DS (differentiated services) field of the received IPv4 header or TC field (traffic class) of IPv6 headers are matched against bits 11:4.

- **UDPM: UDP Port Match**

When UDP port match enable is set (bit 29), the UDP Destination Port of the received UDP frame is matched against bits 27:12.

- **DSTCE: Differentiated Services or Traffic Class Match Enable**

When DS/TC match enable is set (bit 28), the DS (differentiated services) field of the received IPv4 header or TC field (traffic class) of IPv6 headers are matched against bits 11:4.

- **UDPE: UDP Port Match Enable**

When UDP port match enable is set (bit 29), the UDP Destination Port of the received UDP frame is matched against bits 27:12.

Downloaded from Arrow.com.

38.8.114 GMAC Screening Type 2 Register x Priority Queue

Name: GMAC_ST2RPQx[x=0..7]

Address: 0x40050540

Access: Read/Write

31	30	29	28	27	26	25	24
–	COMPCE	COMPC					COMPBE

23	22	21	20	19	18	17	16
COMPB					COMPAE	COMPA	

15	14	13	12	11	10	9	8
COMPA			ETHE	I2ETH			VLANE

7	6	5	4	3	2	1	0
–	VLANP			–	QNB		

Screening type 2 registers are used to allocate up to 3 priority queues to received frames based on the VLAN priority field of received ethernet frames.

- **QNB: Queue Number (0–2)**

If a match is successful, then the queue value programmed in QNB is allocated to the frame.

- **VLANP: VLAN Priority**

When VLAN match enable is set (bit 8), the VLAN priority field of the received frame is matched against bits 7:4 of this register.

- **VLANE: VLAN Enable**

0: VLAN match is disabled.

1: VLAN match is enabled.

- **I2ETH: Index of Screening Type 2 EtherType register x**

When ETHE is set (bit 12), the field EtherType (last EtherType in the header if the frame is VLAN tagged) is compared with bits 15:0 in the register designated by the value of I2ETH.

- **ETHE: EtherType Enable**

0: EtherType match with bits 15:0 in the register designated by the value of I2ETH is disabled.

1: EtherType match with bits 15:0 in the register designated by the value of I2ETH is enabled.

- **COMPA: Index of Screening Type 2 Compare Word 0/Word 1 register x**

COMPA is a pointer to the compare registers GMAC_ST2CW0x and GMAC_ST2CW1x. When COMPAE is set, the compare is true if the data at the frame offset ANDed with the value MASKVAL is equal to the value of COMPVAL ANDed with the value of MASKVAL.

- **COMPAE: Compare A Enable**

0: Comparison via the register designated by index COMPA is disabled.

1: Comparison via the register designated by index COMPA is enabled.

- **COMPB: Index of Screening Type 2 Compare Word 0/Word 1 register x**

Downloaded from Arrow.com.

COMPB is a pointer to the compare registers GMAC_ST2CW0x and GMAC_ST2CW1x. When COMPBE is set, the compare is true if the data at the frame offset ANDed with the value MASKVAL is equal to the value of COMPVAL ANDed with the value of MASKVAL.

- **COMPBE: Compare B Enable**

0: Comparison via the register designated by index COMPB is disabled.

1: Comparison via the register designated by index COMPB is enabled.

- **COMPC: Index of Screening Type 2 Compare Word 0/Word 1 register x**

COMPC is a pointer to the compare registers GMAC_ST2CW0x and GMAC_ST2CW1x. When COMPCE is set, the compare is true if the data at the frame offset ANDed with the value MASKVAL is equal to the value of COMPVAL ANDed with the value of MASKVAL.

- **COMPCE: Compare C Enable**

0: Comparison via the register designated by index COMPC is disabled.

1: Comparison via the register designated by index COMPC is enabled.

Downloaded from Arrow.com.

38.8.115 GMAC Interrupt Enable Register Priority Queue x

Name: GMAC_IERPQx[x=1..2]

Address: 0x40050600

Access: Write-only

31	30	29	28	27	26	25	24
–							

23	22	21	20	19	18	17	16
–							

15	14	13	12	11	10	9	8
–				HRESP	ROVR	–	–

7	6	5	4	3	2	1	0
TCOMP	TFC	RLEX	–	–	RXUBR	RCOMP	–

The following values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

- **RCOMP: Receive Complete**
- **RXUBR: RX Used Bit Read**
- **RLEX: Retry Limit Exceeded or Late Collision**
- **TFC: Transmit Frame Corruption Due to AHB Error**
- **TCOMP: Transmit Complete**
- **ROVR: Receive Overrun**
- **HRESP: HRESP Not OK**

Downloaded from Arrow.com.

38.8.116 GMAC Interrupt Disable Register Priority Queue x

Name: GMAC_IDRPQx[x=1..2]

Address: 0x40050620

Access: Write-only

31	30	29	28	27	26	25	24
–							

23	22	21	20	19	18	17	16
–							

15	14	13	12	11	10	9	8
–				HRESP	ROVR	–	–

7	6	5	4	3	2	1	0
TCOMP	TFC	RLEX	–	–	RXUBR	RCOMP	–

The following values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

- **RCOMP: Receive Complete**
- **RXUBR: RX Used Bit Read**
- **RLEX: Retry Limit Exceeded or Late Collision**
- **TFC: Transmit Frame Corruption Due to AHB Error**
- **TCOMP: Transmit Complete**
- **ROVR: Receive Overrun**
- **HRESP: HRESP Not OK**

Downloaded from Arrow.com.

38.8.117 GMAC Interrupt Mask Register Priority Queue x

Name: GMAC_IMRPQx[x=1..2]

Address: 0x40050640

Access: Read/Write

31	30	29	28	27	26	25	24
–							

23	22	21	20	19	18	17	16
–							

15	14	13	12	11	10	9	8
–				HRESP	ROVR	–	–

7	6	5	4	3	2	1	0
TCOMP	AHB	RLEX	–	–	RXUBR	RCOMP	–

A read of this register returns the value of the receive complete interrupt mask.

A write to this register directly affects the state of the corresponding bit in the Interrupt Status Register, causing an interrupt to be generated if a 1 is written.

The following values are valid for all listed bit names of this register:

0: Corresponding interrupt is enabled.

1: Corresponding interrupt is disabled.

- **RCOMP: Receive Complete**
- **RXUBR: RX Used Bit Read**
- **RLEX: Retry Limit Exceeded or Late Collision**
- **AHB: AHB Error**
- **TCOMP: Transmit Complete**
- **ROVR: Receive Overrun**
- **HRESP: HRESP Not OK**

Downloaded from Arrow.com.

38.8.118 GMAC Screening Type 2 EtherType Register x

Name: GMAC_ST2ERx[x=0..3]

Address: 0x400506E0

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
COMPVAL							
7	6	5	4	3	2	1	0
COMPVAL							

- **COMPVAL: Ethertype Compare Value**

When the bit GMAC_ST2RPQ.ETHE is enabled, the EtherType (last EtherType in the header if the frame is VLAN tagged) is compared with bits 15:0 in the register designated by GMAC_ST2RPQ.I2ETH.

Downloaded from Arrow.com.

38.8.119 GMAC Screening Type 2 Compare Word 0 Register x

Name: GMAC_ST2CW0x[x=0..23]

Access: Read/Write

31	30	29	28	27	26	25	24
COMPVAL							
23	22	21	20	19	18	17	16
COMPVAL							
15	14	13	12	11	10	9	8
MASKVAL							
7	6	5	4	3	2	1	0
MASKVAL							

- **MASKVAL: Mask Value**

The value of MASKVAL ANDed with the 2 bytes extracted from the frame is compared to the value of MASKVAL ANDed with the value of COMPVAL.

- **COMPVAL: Compare Value**

The byte stored in bits [23:16] is compared against the first byte of the 2 bytes extracted from the frame.

The byte stored in bits [31:24] is compared against the second byte of the 2 bytes extracted from the frame.

Downloaded from Arrow.com.

38.8.120 GMAC Screening Type 2 Compare Word 1 Register x

Name: GMAC_ST2CW1x[x=0..23]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	OFFSSTRT

7	6	5	4	3	2	1	0
OFFSSTRT	OFFSVAL						

- **OFFSVAL: Offset Value in Bytes**

The value of OFFSVAL ranges from 0 to 127 bytes, and is counted from either the start of the frame, the byte after the EtherType field (last EtherType in the header if the frame is VLAN tagged), the byte after the IP header (IPv4 or IPv6) or the byte after the TCP/UDP header.

- **OFFSSTRT: Ethernet Frame Offset Start**

Value	Name	Description
0	FRAMESTART	Offset from the start of the frame
1	ETHERTYPE	Offset from the byte after the EtherType field
2	IP	Offset from the byte after the IP header field
3	TCP_UDP	Offset from the byte after the TCP/UDP header field

Downloaded from Arrow.com.

39. High Speed Multimedia Card Interface (HSMCI)

39.1 Description

The High Speed Multimedia Card Interface (HSMCI) supports the MultiMedia Card (MMC) Specification V4.3, the SD Memory Card Specification V2.0, the SDIO V2.0 specification and CE-ATA V1.1.

The HSMCI includes a command register, response registers, data registers, timeout counters and error detection logic that automatically handle the transmission of commands and, when required, the reception of the associated responses and data with a limited processor overhead.

The HSMCI operates at a rate of up to Master Clock divided by 2 and supports the interfacing of 1 slot(s). Each slot may be used to interface with a High Speed MultiMedia Card bus (up to 30 Cards) or with an SD Memory Card. A bit field in the SD Card Register performs this selection.

The SD Memory Card communication is based on a 9-pin interface (clock, command, four data and three power lines) and the High Speed MultiMedia Card on a 7-pin interface (clock, command, one data, three power lines and one reserved for future use).

The SD Memory Card interface also supports High Speed MultiMedia Card operations. The main differences between SD and High Speed MultiMedia Cards are the initialization process and the bus topology.

HSMCI fully supports CE-ATA Revision 1.1, built on the MMC System Specification v4.0. The module includes dedicated hardware to issue the command completion signal and capture the host command completion signal disable.

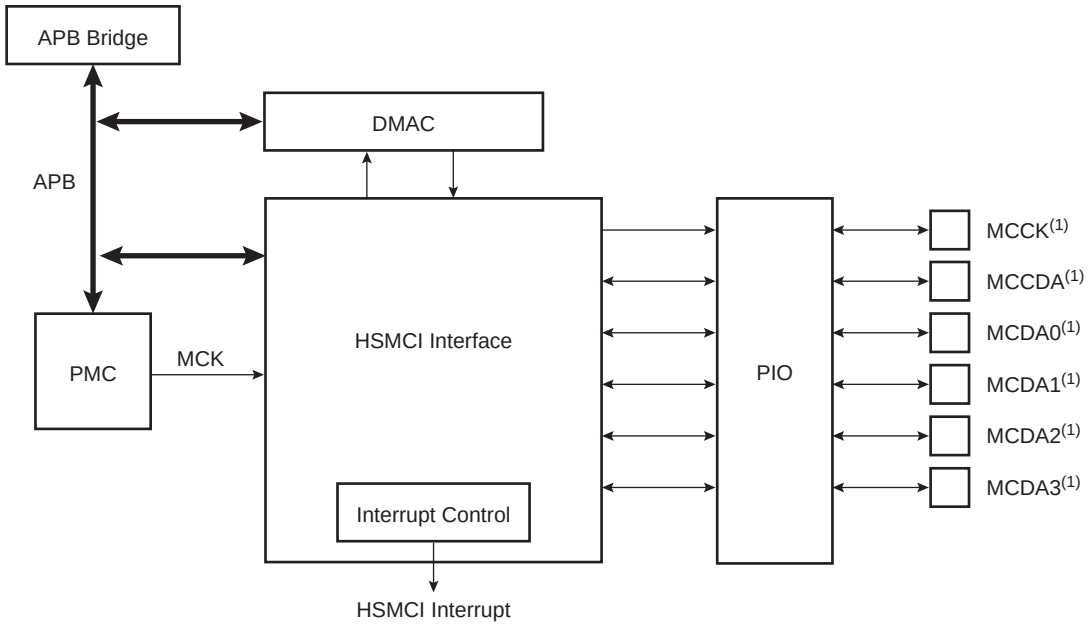
39.2 Embedded Characteristics

- Compatible with MultiMedia Card Specification Version 4.3
- Compatible with SD Memory Card Specification Version 2.0
- Compatible with SDIO Specification Version 2.0
- Compatible with CE-ATA Specification 1.1
- Cards Clock Rate Up to Master Clock Divided by 2
- Boot Operation Mode Support
- High Speed Mode Support
- Embedded Power Management to Slow Down Clock Rate When Not Used
- Supports 1 Multiplexed Slot(s)
  - Each Slot for either a High Speed MultiMedia Card Bus (Up to 30 Cards) or an SD Memory Card
- Support for Stream, Block and Multi-block Data Read and Write
  - Minimizes Processor Intervention for Large Buffer Transfers
- Built in FIFO (from 16 to 256 bytes) with Large Memory Aperture Supporting Incremental Access
- Support for CE-ATA Completion Signal Disable Command
- Protection Against Unexpected Modification On-the-Fly of the Configuration Registers

Downloaded from Arrow.com.

39.3 Block Diagram

Figure 39-1. Block Diagram (4-bit configuration)

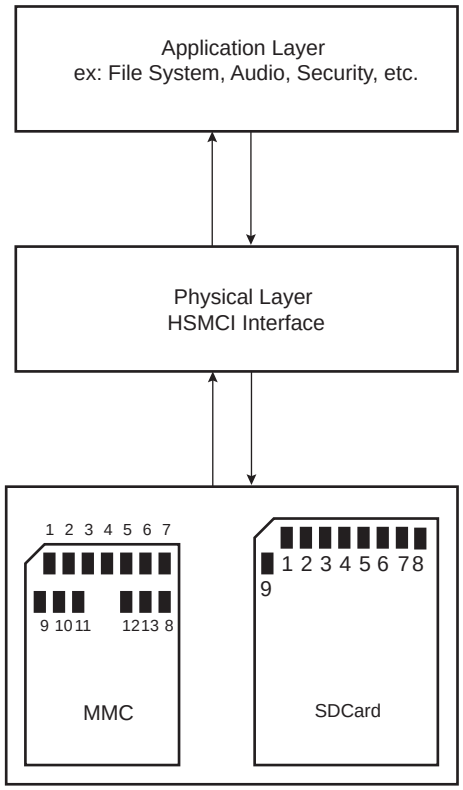


Note: 1. When several HSMCI (x HSMCI) are embedded in a product, MCCK refers to HSMCIx_CK, MCCDA to HSMCIx_CDA, MCDAy to HSMCIx_DAy.

Downloaded from Arrow.com.

39.4 Application Block Diagram

Figure 39-2. Application Block Diagram



39.5 Pin Name List

Table 39-1. I/O Lines Description for 4-bit Configuration

Pin Name[1]	Pin Description	Type[2]	Comments
MCCDA	Command/response	I/O/PP/OD	CMD of an MMC or SDCard/SDIO
MCCK	Clock	I/O	CLK of an MMC or SD Card/SDIO
MCDA0–MCDA3	Data 0..3 of Slot A	I/O/PP	DAT[0..3] of an MMC DAT[0..3] of an SD Card/SDIO

Notes:
1. When several HSMCI (x HSMCI) are embedded in a product, MCCK refers to HSMCIx_CK, MCCDA to HSMCIx_CDA, MCDAy to HSMCIx_DAy.
2. I: Input, O: Output, PP: Push/Pull, OD: Open Drain.

Downloaded from Arrow.com.

39.6 Product Dependencies

39.6.1 I/O Lines

The pins used for interfacing the High Speed MultiMedia Cards or SD Cards are multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the peripheral functions to HSMCI pins.

Table 39-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
HSMCI	MCCDA	PA28	C
HSMCI	MCCK	PA25	D
HSMCI	MCDA0	PA30	C
HSMCI	MCDA1	PA31	C
HSMCI	MCDA2	PA26	C
HSMCI	MCDA3	PA27	C

39.6.2 Power Management

The HSMCI is clocked through the Power Management Controller (PMC), so the programmer must first configure the PMC to enable the HSMCI clock.

39.6.3 Interrupt Sources

The HSMCI has an interrupt line connected to the interrupt controller.

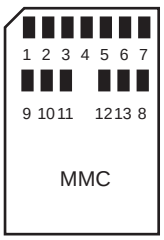
Handling the HSMCI interrupt requires programming the interrupt controller before configuring the HSMCI.

Table 39-3. Peripheral IDs

Instance	ID
HSMCI	18

39.7 Bus Topology

Figure 39-3. High Speed MultiMedia Memory Card Bus Topology



Downloaded from Arrow.com.

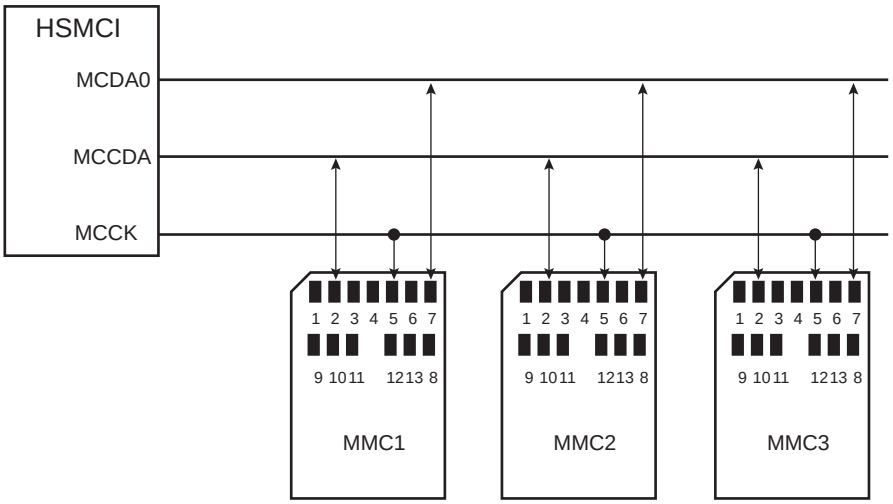
The High Speed MultiMedia Card communication is based on a 13-pin serial bus interface. It has three communication lines and four supply lines.

Table 39-4. Bus Topology

Pin Number	Name	Type[1]	Description	HSMCI Pin Name[2] (Slot z)
1	DAT[3]	I/O/PP	Data	MCDz3
2	CMD	I/O/PP/OD	Command/response	MCCDz
3	VSS1	S	Supply voltage ground	VSS
4	VDD	S	Supply voltage	VDD
5	CLK	I/O	Clock	MCCK
6	VSS2	S	Supply voltage ground	VSS
7	DAT[0]	I/O/PP	Data 0	MCDz0
8	DAT[1]	I/O/PP	Data 1	MCDz1
9	DAT[2]	I/O/PP	Data 2	MCDz2

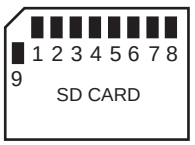
Notes: 1. I: Input, O: Output, PP: Push/Pull, OD: Open Drain.
2. When several HSMCI (x HSMCI) are embedded in a product, MCCK refers to HSMCIx_CK, MCCDA to HSMCIx_CDA, MCDAy to HSMCIx_DAy.

Figure 39-4. MMC Bus Connections (One Slot)



Note: When several HSMCI (x HSMCI) are embedded in a product, MCCK refers to HSMCIx_CK, MCCDA to HSMCIx_CDA MCDAy to HSMCIx_DAy.

Figure 39-5. SD Memory Card Bus Topology



Downloaded from Arrow.com.

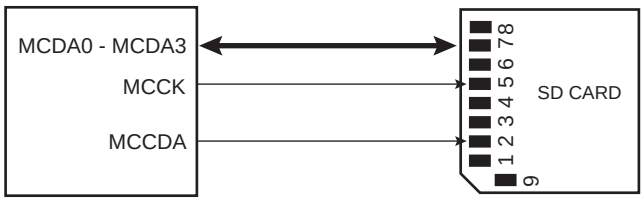
The SD Memory Card bus includes the signals listed in Table 39-5.

Table 39-5. SD Memory Card Bus Signals

Pin Number	Name	Type[1]	Description	HSMCI Pin Name[2] (Slot z)
1	CD/DAT[3]	I/O/PP	Card detect/ Data line Bit 3	MCDz3
2	CMD	PP	Command/response	MCCDz
3	VSS1	S	Supply voltage ground	VSS
4	VDD	S	Supply voltage	VDD
5	CLK	I/O	Clock	MCCK
6	VSS2	S	Supply voltage ground	VSS
7	DAT[0]	I/O/PP	Data line Bit 0	MCDz0
8	DAT[1]	I/O/PP	Data line Bit 1 or Interrupt	MCDz1
9	DAT[2]	I/O/PP	Data line Bit 2	MCDz2

Notes: 1. I: input, O: output, PP: Push Pull, OD: Open Drain.
2. When several HSMCI (x HSMCI) are embedded in a product, MCCK refers to HSMCIx_CK, MCCDA to HSMCIx_CDA, MCDAy to HSMCIx_DAy.

Figure 39-6. SD Card Bus Connections with One Slot



Note: When several HSMCI (x HSMCI) are embedded in a product, MCCK refers to HSMCIx_CK, MCCDA to HSMCIx_CDA MCDAy to HSMCIx_DAy.

When the HSMCI is configured to operate with SD memory cards, the width of the data bus can be selected in the HSMCI_SDCR. Clearing the SDCBUS bit in this register means that the width is one bit; setting it means that the width is four bits. In the case of High Speed MultiMedia cards, only the data line 0 is used. The other data lines can be used as independent PIOs.

Downloaded from Arrow.com.

39.8 High Speed MultiMedia Card Operations

After a power-on reset, the cards are initialized by a special message-based High Speed MultiMedia Card bus protocol. Each message is represented by one of the following tokens:

- Command—A command is a token that starts an operation. A command is sent from the host either to a single card (addressed command) or to all connected cards (broadcast command). A command is transferred serially on the CMD line.
- Response—A response is a token which is sent from an addressed card or (synchronously) from all connected cards to the host as an answer to a previously received command. A response is transferred serially on the CMD line.
- Data—Data can be transferred from the card to the host or vice versa. Data is transferred via the data line.

Card addressing is implemented using a session address assigned during the initialization phase by the bus controller to all currently connected cards. Their unique CID number identifies individual cards.

The structure of commands, responses and data blocks is described in the High Speed MultiMedia Card System Specification. See also Table 39-6 on page 880.

High Speed MultiMedia Card bus data transfers are composed of these tokens.

There are different types of operations. Addressed operations always contain a command and a response token. In addition, some operations have a data token; the others transfer their information directly within the command or response structure. In this case, no data token is present in an operation. The bits on the DAT and the CMD lines are transferred synchronous to the clock HSMCI clock.

Two types of data transfer commands are defined:

- Sequential commands—These commands initiate a continuous data stream. They are terminated only when a stop command follows on the CMD line. This mode reduces the command overhead to an absolute minimum.
- Block-oriented commands—These commands send a data block succeeded by CRC bits.

Both read and write operations allow either single or multiple block transmission. A multiple block transmission is terminated when a stop command follows on the CMD line similarly to the sequential read or when a multiple block transmission has a predefined block count (see Section 39.8.2 “Data Transfer Operation”).

The HSMCI provides a set of registers to perform the entire range of High Speed MultiMedia Card operations.

39.8.1 Command - Response Operation

After reset, the HSMCI is disabled and becomes valid after setting the MCIEN bit in the HSMCI_CR.

The PWSEN bit saves power by dividing the HSMCI clock by $2^{PWSDIV} + 1$ when the bus is inactive.

The two bits, RDPROOF and WRPROOF in the HSMCI Mode Register (HSMCI_MR) allow stopping the HSMCI clock during read or write access if the internal FIFO is full. This will guarantee data integrity, not bandwidth.

All the timings for High Speed MultiMedia Card are defined in the High Speed MultiMedia Card System Specification.

The two bus modes (open drain and push/pull) needed to process all the operations are defined in the HSMCI Command Register (HSMCI_CMDR). The HSMCI_CMDR allows a command to be carried out.

For example, to perform an ALL_SEND_CID command:

	Host Command					N_{ID} Cycles			Response			High Impedance State		
CMD	S	T	Content	CRC	E	Z	******	Z	S	T	CID Content	Z	Z	Z

Downloaded from Arrow.com.

The command ALL_SEND_CID and the fields and values for the HSMCI_CMDR are described in Table 39-6 and Table 39-7.

Table 39-6. ALL_SEND_CID Command Description

CMD Index	Type	Argument	Response	Abbreviation	Command Description
CMD2	bcr[1]	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the CMD line

Note: 1. bcr means broadcast command with response.

Table 39-7. Fields and Values for HSMCI_CMDR

Field	Value
CMDNB (command number)	2 (CMD2)
RSPTYP (response type)	2 (R2: 136 bits response)
SPCMD (special command)	0 (not a special command)
OPCMD (open drain command)	1
MAXLAT (max latency for command to response)	0 (NID cycles ==> 5 cycles)
TRCMD (transfer command)	0 (No transfer)
TRDIR (transfer direction)	X (available only in transfer command)
TRTYP (transfer type)	X (available only in transfer command)
IOSPCMD (SDIO special command)	0 (not a special command)

The HSMCI_ARGR contains the argument field of the command.

To send a command, the user must perform the following steps:

- Fill the argument register (HSMCI_ARGR) with the command argument.
- Set the command register (HSMCI_CMDR) (see Table 39-7).

The command is sent immediately after writing the command register.

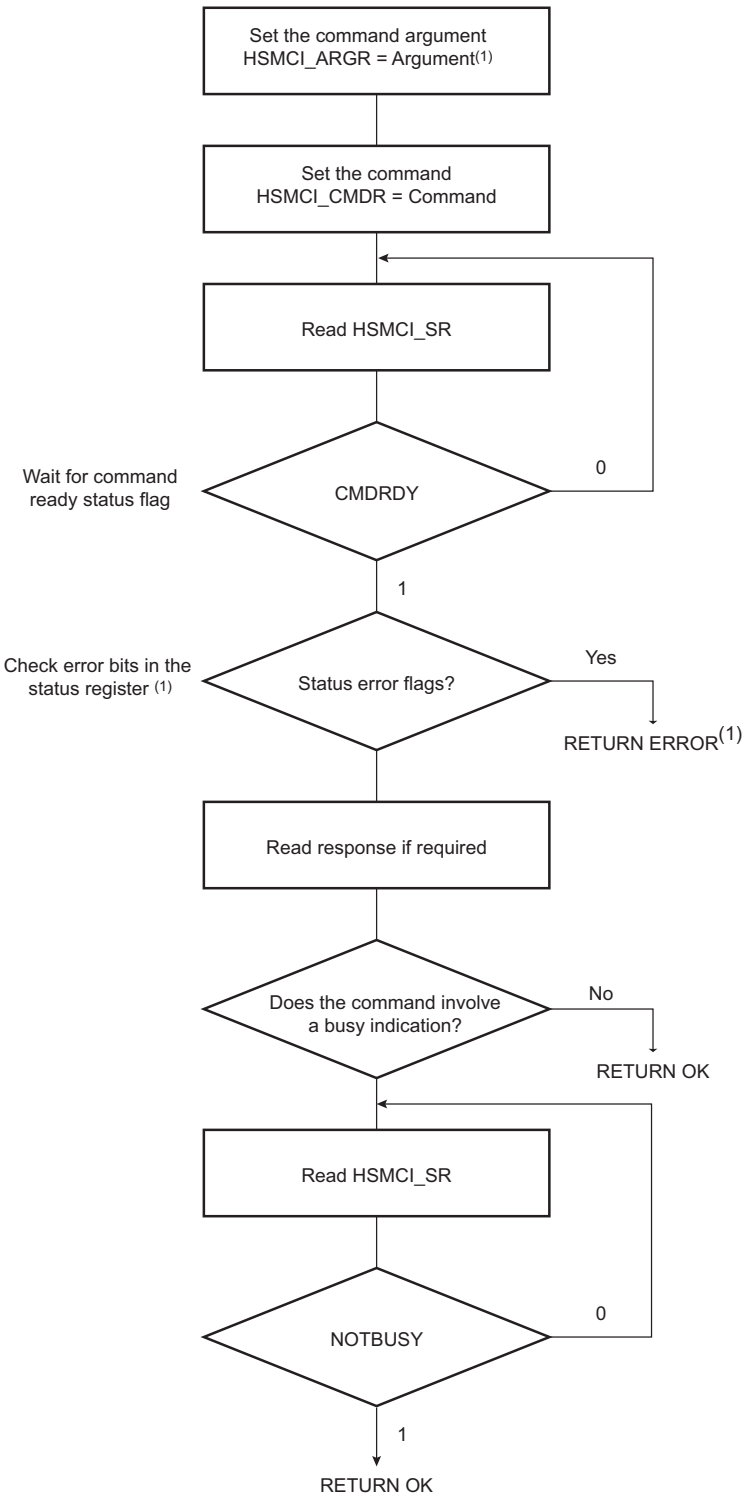
While the card maintains a busy indication (at the end of a STOP_TRANSMISSION command CMD12, for example), a new command shall not be sent. The NOTBUSY flag in the Status Register (HSMCI_SR) is asserted when the card releases the busy indication.

If the command requires a response, it can be read in the HSMCI Response Register (HSMCI_RSPR). The response size can be from 48 bits up to 136 bits depending on the command. The HSMCI embeds an error detection to prevent any corrupted data during the transfer.

The following flowchart shows how to send a command to the card and read the response if needed. In this example, the status register bits are polled but setting the appropriate bits in the HSMCI Interrupt Enable Register (HSMCI_IER) allows using an interrupt method.

Downloaded from Arrow.com.

Figure 39-7. Command/Response Functional Flow Diagram



Note: If the command is SEND_OP_COND, the CRC error flag is always present (refer to R3 response in the High Speed MultiMedia Card specification).

Downloaded from Arrow.com.

39.8.2 Data Transfer Operation

The High Speed MultiMedia Card allows several read/write operations (single block, multiple blocks, stream, etc.). These kinds of transfer can be selected setting the Transfer Type (TRTYP) field in the HSMCI Command Register (HSMCI_CMDR).

In all cases, the block length (BLKLEN field) must be defined either in the HSMCI Mode Register (HSMCI_MR) or in the HSMCI Block Register (HSMCI_BLKR). This field determines the size of the data block.

Consequent to MMC Specification 3.1, two types of multiple block read (or write) transactions are defined (the host can use either one at any time):

- Open-ended/Infinite Multiple block read (or write):

  The number of blocks for the read (or write) multiple block operation is not defined. The card will continuously transfer (or program) data blocks until a stop transmission command is received.

- Multiple block read (or write) with predefined block count (since version 3.1 and higher):

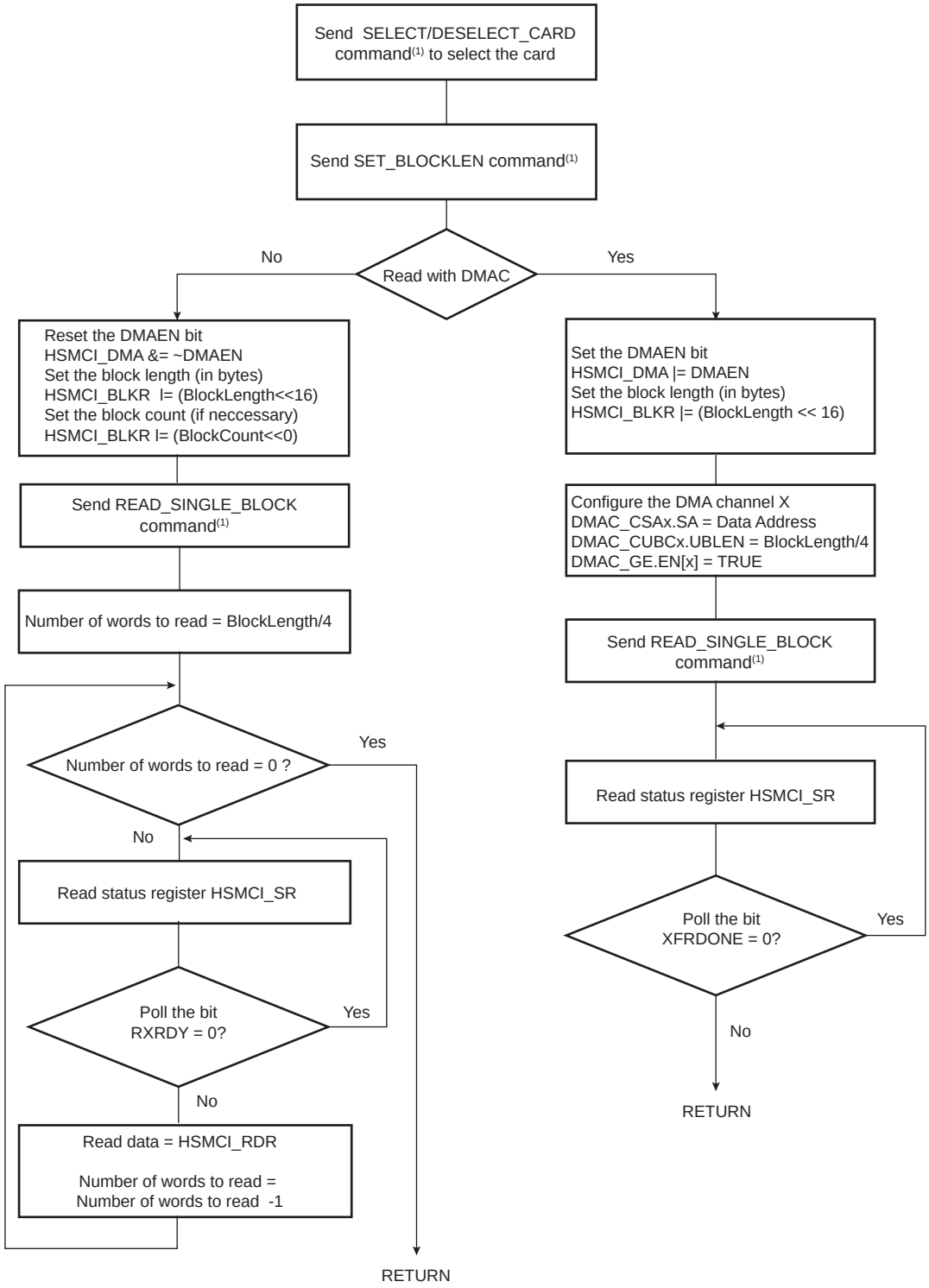
  The card will transfer (or program) the requested number of data blocks and terminate the transaction. The stop command is not required at the end of this type of multiple block read (or write), unless terminated with an error. In order to start a multiple block read (or write) with predefined block count, the host must correctly program the HSMCI Block Register (HSMCI_BLKR). Otherwise the card will start an open-ended multiple block read. The BCNT field of the HSMCI_BLKR defines the number of blocks to transfer (from 1 to 65535 blocks). Programming the value 0 in the BCNT field corresponds to an infinite block transfer.

39.8.3 Read Operation

The following flowchart (Figure 39-8) shows how to read a single block with or without use of DMAC facilities. In this example, a polling method is used to wait for the end of read. Similarly, the user can configure the HSMCI Interrupt Enable Register (HSMCI_IER) to trigger an interrupt at the end of read.

Downloaded from Arrow.com.

Figure 39-8. Read Functional Flow Diagram



Notes: 1. It is assumed that this command has been correctly sent (see Figure 39-7).

Downloaded from Arrow.com.

39.8.4 Write Operation

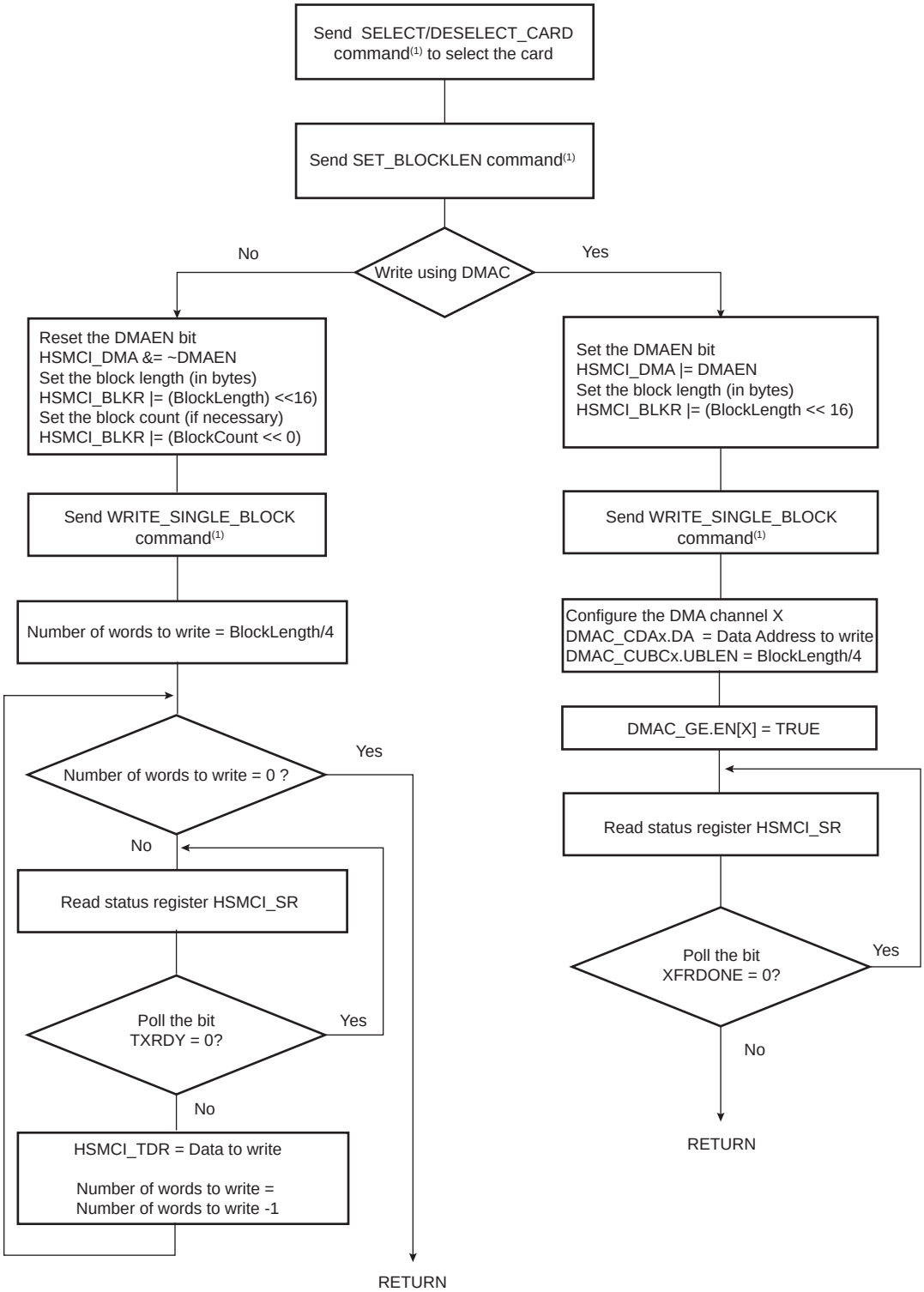
In write operation, the HSMCI Mode Register (HSMCI_MR) is used to define the padding value when writing non-multiple block size. If the bit PADV is 0, then 0x00 value is used when padding data, otherwise 0xFF is used.

If set, the bit DMAEN in the HSMCI DMA Condiguration Register (HSMCI_DMA) enables DMA transfer.

The flowchart in Figure 39-9 shows how to write a single block with or without use of DMA facilities. Polling or interrupt method can be used to wait for the end of write according to the contents of the HSMCI Interrupt Mask Register (HSMCI_IMR).

Downloaded from Arrow.com.

Figure 39-9. Write Functional Flow Diagram

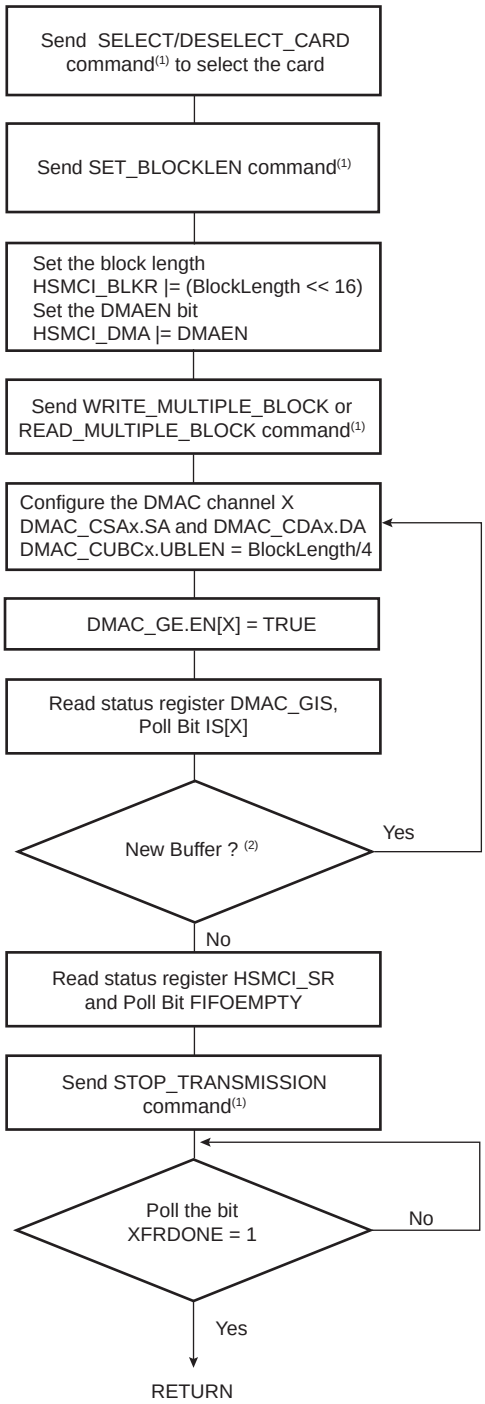


Notes: 1. It is assumed that this command has been correctly sent (see Figure 39-7).

Downloaded from Arrow.com.

The flowchart in Figure 39-10 shows how to manage read multiple block and write multiple block transfers with the DMA Controller. Polling or interrupt method can be used to wait for the end of write according to the contents of the HSMCI_IMR.

Figure 39-10. Read and Write Multiple Block



Notes: 1. It is assumed that this command has been correctly sent (see Figure 39-7).
2. Handle errors reported in HSMCI_SR.

Downloaded from Arrow.com.

39.8.5 WRITE_SINGLE_BLOCK/WRITE_MULTIPLE_BLOCK Operation using DMA Controller

1. Wait until the current command execution has successfully terminated.
   a. Check that CMDRDY and NOTBUSY fields are asserted in HSMCI_SR
2. Program the block length in the card. This value defines the value *block_length.*
3. Program the block length in the HSMCI Configuration Register with *block_length* value.
4. Configure the fields of the HSMCI_MR as follows:
   a. Program FBYTE to one when the transfer is not multiple of 4, zero otherwise.
5. Issue a WRITE_SINGLE_BLOCK command writing HSMCI_ARG then HSMCI_CMDR.
6. Program the DMA Controller.
   a. Read the Channel Status Register to choose an available (disabled) channel.
   b. Clear any pending interrupts on the channel from the previous DMAC transfer by reading the DMAC_CISx register.
   c. Program the channel registers.
   d. The DMAC_CSAx register for Channel x must be set to the location of the source data.
   e. The DMAC_CDAx register for Channel x must be set with the starting address of the HSMCI_FIFO address.
   f. Configure the fields of DMAC_CCx of Channel x as follows:
      –DWIDTH is set to WORD when the transfer is multiple of 4, otherwise it is set to BYTE
      –CSIZE must be set according to the value of HSMCI_DMA.CHKSIZE.
   g. Configure the fields of DMAC_CUBCx for Channel x as follows:
      –UBLEN is programmed with *block_length/4* when the transfer length is multiple of 4, *block_length* otherwise.
   h. Enable Channel x, writing one to DMAC_GE.EN[x]. The DMAC is ready and waiting for request.
7. Wait for XFRDONE in the HSMCI_SR.

39.8.6 READ_SINGLE_BLOCK/READ_MULTIPLE_BLOCK Operation using DMA Controller

1. Wait until the current command execution has successfully completed.
   a. Check that CMDRDY and NOTBUSY are asserted in HSMCI_SR.
2. Program the block length in the card. This value defines the value *block_length.*
3. Program the block length in the HSMCI Configuration Register with *block_length* value.
4. Set RDPROOF bit in HSMCI_MR to avoid overflow.
5. Configure the fields of the HSMCI_MR as follows:
   a. Program FBYTE to one when the transfer is not multiple of 4, zero otherwise.
6. Issue a READ_SINGLE_BLOCK/WRITE_MULTIPLE_BLOCK command.
7. Program the DMA controller.
   a. Read the Channel Status Register to choose an available (disabled) channel.
   b. Clear any pending interrupts on the channel from the previous DMA transfer by reading the DMAC_CISx register.
   c. Program the channel registers.
   d. The DMAC_CSAx register for Channel x must be set with the starting address of the HSMCI_FIFO address.
   e. The DMAC_CDAx register for Channel x must be word aligned.
   f. Configure the fields of DMAC_CCx for Channel x as follows:
      –DWIDTH is set to WORD when the length is a multiple of 4, otherwise it is set to BYTE.
      –CSIZE must be set according to the value of HSMCI_DMA.CHKSIZE.

Downloaded from Arrow.com.

g. Configure the fields of the DMAC_CUBCx register of Channel x as follows:
   –UBLEN is programmed with *block_length/4* when the transfer length is multiple of 4, *block_length* otherwise.
h. Enable Channel x, writing one to DMAC_GE.EN[x]. The DMAC is ready and waiting for request.

8. Wait for XFRDONE in the HSMCI_SR.

39.9 SD/SDIO Card Operation

The High Speed MultiMedia Card Interface allows processing of SD Memory (Secure Digital Memory Card) and SDIO (SD Input Output) Card commands.

SD/SDIO cards are based on the MultiMedia Card (MMC) format, but are physically slightly thicker and feature higher data transfer rates, a lock switch on the side to prevent accidental overwriting and security features. The physical form factor, pin assignment and data transfer protocol are forward-compatible with the High Speed MultiMedia Card with some additions. SD slots can actually be used for more than flash memory cards. Devices that support SDIO can use small devices designed for the SD form factor, such as GPS receivers, Wi-Fi or Bluetooth adapters, modems, barcode readers, IrDA adapters, FM radio tuners, RFID readers, digital cameras and more.

SD/SDIO is covered by numerous patents and trademarks, and licensing is only available through the Secure Digital Card Association.

The SD/SDIO Card communication is based on a 9-pin interface (Clock, Command, 4 x Data and 3 x Power lines). The communication protocol is defined as a part of this specification. The main difference between the SD/SDIO Card and the High Speed MultiMedia Card is the initialization process.

The SD/SDIO Card Register (HSMCI_SDCR) allows selection of the Card Slot and the data bus width.

The SD/SDIO Card bus allows dynamic configuration of the number of data lines. After power up, by default, the SD/SDIO Card uses only DAT0 for data transfer. After initialization, the host can change the bus width (number of active data lines).

39.9.1 SDIO Data Transfer Type

SDIO cards may transfer data in either a multi-byte (1 to 512 bytes) or an optional block format (1 to 511 blocks), while the SD memory cards are fixed in the block transfer mode. The TRTYP field in the HSMCI Command Register (HSMCI_CMDR) allows to choose between SDIO Byte or SDIO Block transfer.

The number of bytes/blocks to transfer is set through the BCNT field in the HSMCI Block Register (HSMCI_BLKR). In SDIO Block mode, the field BLKLEN must be set to the data block size while this field is not used in SDIO Byte mode.

An SDIO Card can have multiple I/O or combined I/O and memory (called Combo Card). Within a multi-function SDIO or a Combo card, there are multiple devices (I/O and memory) that share access to the SD bus. In order to allow the sharing of access to the host among multiple devices, SDIO and combo cards can implement the optional concept of suspend/resume (Refer to the SDIO Specification for more details). To send a suspend or a resume command, the host must set the SDIO Special Command field (IOSPCMD) in the HSMCI Command Register.

39.9.2 SDIO Interrupts

Each function within an SDIO or Combo card may implement interrupts (Refer to the SDIO Specification for more details). In order to allow the SDIO card to interrupt the host, an interrupt function is added to a pin on the DAT[1] line to signal the card's interrupt to the host. An SDIO interrupt on each slot can be enabled through the HSMCI Interrupt Enable Register. The SDIO interrupt is sampled regardless of the currently selected slot.

Downloaded from Arrow.com.

39.10 CE-ATA Operation

CE-ATA maps the streamlined ATA command set onto the MMC interface. The ATA task file is mapped onto MMC register space.

CE-ATA utilizes five MMC commands:

- GO_IDLE_STATE (CMD0): used for hard reset.
- STOP_TRANSMISSION (CMD12): causes the ATA command currently executing to be aborted.
- FAST_IO (CMD39): Used for single register access to the ATA taskfile registers, 8-bit access only.
- RW_MULTIPLE_REGISTERS (CMD60): used to issue an ATA command or to access the control/status registers.
- RW_MULTIPLE_BLOCK (CMD61): used to transfer data for an ATA command.

CE-ATA utilizes the same MMC command sequences for initialization as traditional MMC devices.

39.10.1 Executing an ATA Polling Command

1. Issue READ_DMA_EXT with RW_MULTIPLE_REGISTER (CMD60) for 8 KB of DATA.
2. Read the ATA status register until DRQ is set.
3. Issue RW_MULTIPLE_BLOCK (CMD61) to transfer DATA.
4. Read the ATA status register until DRQ && BSY are configured to 0.

39.10.2 Executing an ATA Interrupt Command

1. Issue READ_DMA_EXT with RW_MULTIPLE_REGISTER (CMD60) for 8 KB of DATA with nIEN field set to zero to enable the command completion signal in the device.
2. Issue RW_MULTIPLE_BLOCK (CMD61) to transfer DATA.
3. Wait for Completion Signal Received Interrupt.

39.10.3 Aborting an ATA Command

If the host needs to abort an ATA command prior to the completion signal it must send a special command to avoid potential collision on the command line. The SPCMD field of the HSMCI_CMDR must be set to 3 to issue the CE-ATA completion Signal Disable Command.

39.10.4 CE-ATA Error Recovery

Several methods of ATA command failure may occur, including:

- No response to an MMC command, such as RW_MULTIPLE_REGISTER (CMD60).
- CRC is invalid for an MMC command or response.
- CRC16 is invalid for an MMC data packet.
- ATA Status register reflects an error by setting the ERR bit to one.
- The command completion signal does not arrive within a host specified time out period.

Error conditions are expected to happen infrequently. Thus, a robust error recovery mechanism may be used for each error event. The recommended error recovery procedure after a timeout is:

- Issue the command completion signal disable if nIEN was cleared to zero and the RW_MULTIPLE_BLOCK (CMD61) response has been received.
- Issue STOP_TRANSMISSION (CMD12) and successfully receive the R1 response.
- Issue a software reset to the CE-ATA device using FAST_IO (CMD39).

If STOP_TRANMISSION (CMD12) is successful, then the device is again ready for ATA commands. However, if the error recovery procedure does not work as expected or there is another timeout, the next step is to issue

Downloaded from Arrow.com.

GO_IDLE_STATE (CMD0) to the device. GO_IDLE_STATE (CMD0) is a hard reset to the device and completely resets all device states.

Note that after issuing GO_IDLE_STATE (CMD0), all device initialization needs to be completed again. If the CE-ATA device completes all MMC commands correctly but fails the ATA command with the ERR bit set in the ATA Status register, no error recovery action is required. The ATA command itself failed implying that the device could not complete the action requested, however, there was no communication or protocol failure. After the device signals an error by setting the ERR bit to one in the ATA Status register, the host may attempt to retry the command.

39.11 HSMCI Boot Operation Mode

In boot operation mode, the processor can read boot data from the slave (MMC device) by keeping the CMD line low after power-on before issuing CMD1. The data can be read from either the boot area or user area, depending on register setting.

39.11.1 Boot Procedure, Processor Mode

1. Configure the HSMCI data bus width programming SDCBUS Field in the HSMCI_SDCR. The BOOT_BUS_WIDTH field located in the device Extended CSD register must be set accordingly.
2. Set the byte count to 512 bytes and the block count to the desired number of blocks, writing BLKLEN and BCNT fields of the HSMCI_BLKR.
3. Issue the Boot Operation Request command by writing to the HSMCI_CMDR with SPCMD field set to BOOTREQ, TRDIR set to READ and TRCMD set to “start data transfer”.
4. The BOOT_ACK field located in the HSMCI_CMDR must be set to one, if the BOOT_ACK field of the MMC device located in the Extended CSD register is set to one.
5. Host processor can copy boot data sequentially as soon as the RXRDY flag is asserted.
6. When Data transfer is completed, host processor shall terminate the boot stream by writing the HSMCI_CMDR with SPCMD field set to BOOTEND.

39.11.2 Boot Procedure DMA Mode

1. Configure the HSMCI data bus width by programming SDCBUS Field in the HSMCI_SDCR. The BOOT_BUS_WIDTH field in the device Extended CSD register must be set accordingly.
2. Set the byte count to 512 bytes and the block count to the desired number of blocks by writing BLKLEN and BCNT fields of the HSMCI_BLKR.
3. Enable DMA transfer in the HSMCI_DMA register.
4. Configure DMA controller, program the total amount of data to be transferred and enable the relevant channel.
5. Issue the Boot Operation Request command by writing to the HSMCI_CMDR with SPCND set to BOOTREQ, TRDIR set to READ and TRCMD set to “start data transfer”.
6. DMA controller copies the boot partition to the memory.
7. When DMA transfer is completed, host processor shall terminate the boot stream by writing the HSMCI_CMDR with SPCMD field set to BOOTEND.

Downloaded from Arrow.com.

39.12 HSMCI Transfer Done Timings

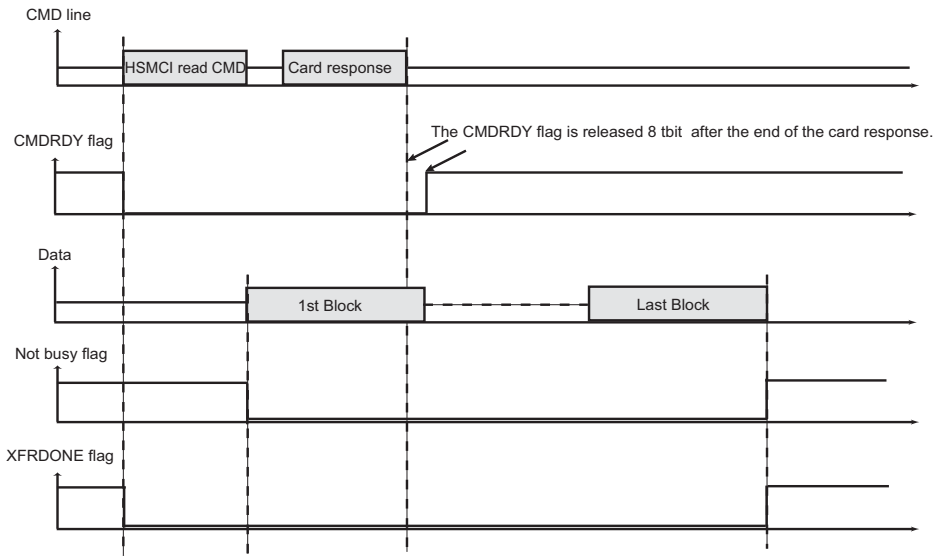
39.12.1 Definition

The XFRDONE flag in the HSMCI_SR indicates exactly when the read or write sequence is finished.

39.12.2 Read Access

During a read access, the XFRDONE flag behaves as shown in Figure 39-11.

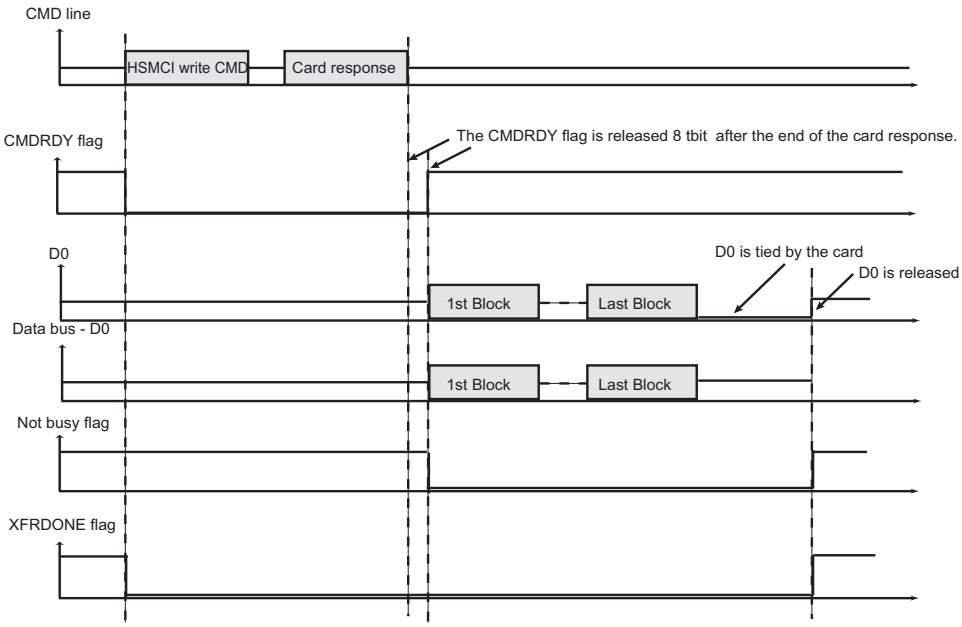
Figure 39-11. XFRDONE During a Read Access



39.12.3 Write Access

During a write access, the XFRDONE flag behaves as shown in Figure 39-12.

Figure 39-12. XFRDONE During a Write Access



Downloaded from Arrow.com.

39.13 Register Write Protection

To prevent any single software error from corrupting HSMCI behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the HSMCI Write Protection Mode Register (HSMCI_WPMR).

If a write access to a write-protected register is detected, the WPVS bit in the HSMCI Write Protection Status Register (HSMCI_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading the HSMCI_WPSR.

The following registers can be protected:

- HSMCI Mode Register
- HSMCI Data Timeout Register
- HSMCI SDCard/SDIO Register
- HSMCI Completion Signal Timeout Register
- HSMCI DMA Configuration Register
- HSMCI Configuration Register

Downloaded from Arrow.com.

39.14 High Speed MultiMedia Card Interface (HSMCI) User Interface

Table 39-8. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	HSMCI_CR	Write-only	–
0x04	Mode Register	HSMCI_MR	Read/Write	0x0
0x08	Data Timeout Register	HSMCI_DTOR	Read/Write	0x0
0x0C	SD/SDIO Card Register	HSMCI_SDCR	Read/Write	0x0
0x10	Argument Register	HSMCI_ARGR	Read/Write	0x0
0x14	Command Register	HSMCI_CMDR	Write-only	–
0x18	Block Register	HSMCI_BLKR	Read/Write	0x0
0x1C	Completion Signal Timeout Register	HSMCI_CSTOR	Read/Write	0x0
0x20	Response Register(1)	HSMCI_RSPR	Read-only	0x0
0x24	Response Register(1)	HSMCI_RSPR	Read-only	0x0
0x28	Response Register(1)	HSMCI_RSPR	Read-only	0x0
0x2C	Response Register(1)	HSMCI_RSPR	Read-only	0x0
0x30	Receive Data Register	HSMCI_RDR	Read-only	0x0
0x34	Transmit Data Register	HSMCI_TDR	Write-only	–
0x38–0x3C	Reserved	–	–	–
0x40	Status Register	HSMCI_SR	Read-only	0xC0E5
0x44	Interrupt Enable Register	HSMCI_IER	Write-only	–
0x48	Interrupt Disable Register	HSMCI_IDR	Write-only	–
0x4C	Interrupt Mask Register	HSMCI_IMR	Read-only	0x0
0x50	DMA Configuration Register	HSMCI_DMA	Read/Write	0x00
0x54	Configuration Register	HSMCI_CFG	Read/Write	0x00
0x58–0xE0	Reserved	–	–	–
0xE4	Write Protection Mode Register	HSMCI_WPMR	Read/Write	–
0xE8	Write Protection Status Register	HSMCI_WPSR	Read-only	–
0xEC–0xFC	Reserved	–	–	–
0x100–0x1FC	Reserved	–	–	–
0x200	FIFO Memory Aperture0	HSMCI_FIFO0	Read/Write	0x0
...	...	...	...	...
0x5FC	FIFO Memory Aperture255	HSMCI_FIFO255	Read/Write	0x0

Notes: 1. The Response Register can be read by N accesses at the same HSMCI_RSPR or at consecutive addresses (0x20 to 0x2C). N depends on the size of the response.

Downloaded from Arrow.com.

39.14.1 HSMCI Control Register

Name: HSMCI_CR

Address: 0x40000000

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
SWRST	–	–	–	PWSDIS	PWSEN	MCIDIS	MCIEN

- **MCIEN: Multi-Media Interface Enable**

0: No effect.

1: Enables the Multi-Media Interface if MCDIS is 0.

- **MCIDIS: Multi-Media Interface Disable**

0: No effect.

1: Disables the Multi-Media Interface.

- **PWSEN: Power Save Mode Enable**

0: No effect.

1: Enables the Power Saving Mode if PWSDIS is 0.

Warning: Before enabling this mode, the user must set a value different from 0 in the PWSDIV field of the HSMCI_MR.

- **PWSDIS: Power Save Mode Disable**

0: No effect.

1: Disables the Power Saving Mode.

- **SWRST: Software Reset**

0: No effect.

1: Resets the HSMCI. A software triggered hardware reset of the HSMCI is performed.

Downloaded from Arrow.com.

39.14.2 HSMCI Mode Register

Name: HSMCI_MR

Address: 0x40000004

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	CLKODD
15	14	13	12	11	10	9	8
–	PADV	FBYTE	WRPROOF	RDPROOF	PWSDIV		
7	6	5	4	3	2	1	0
CLKDIV							

This register can only be written if the WPEN bit is cleared in the HSMCI Write Protection Mode Register.

- **CLKDIV: Clock Divider**

High Speed MultiMedia Card Interface clock (MCCK or HSMCI_CK) is Master Clock (MCK) divided by ({CLKDIV,CLKODD}+2).

- **PWSDIV: Power Saving Divider**

High Speed MultiMedia Card Interface clock is divided by $2^{(PWSDIV)}$ + 1 when entering Power Saving Mode.

Warning: This value must be different from 0 before enabling the Power Save Mode in the HSMCI_CR (HSMCI_PWSEN bit).

- **RDPROOF: Read Proof Enable**

Enabling Read Proof allows to stop the HSMCI Clock during read access if the internal FIFO is full. This will guarantee data integrity, not bandwidth.

0: Disables Read Proof.

1: Enables Read Proof.

- **WRPROOF: Write Proof Enable**

Enabling Write Proof allows to stop the HSMCI Clock during write access if the internal FIFO is full. This will guarantee data integrity, not bandwidth.

0: Disables Write Proof.

1: Enables Write Proof.

- **FBYTE: Force Byte Transfer**

Enabling Force Byte Transfer allow byte transfers, so that transfer of blocks with a size different from modulo 4 can be supported.

Warning: BLKLEN value depends on FBYTE.

0: Disables Force Byte Transfer.

1: Enables Force Byte Transfer.

Downloaded from Arrow.com.

- **PADV: Padding Value**

0: 0x00 value is used when padding data in write transfer.

1: 0xFF value is used when padding data in write transfer.

PADV may be only in manual transfer.

- **CLKODD: Clock divider is odd**

This bit is the least significant bit of the clock divider and indicates the clock divider parity.

Downloaded from Arrow.com.

39.14.3 HSMCI Data Timeout Register

Name: HSMCI_DTOR

Address: 0x40000008

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	DTOMUL			DTOCYC			

This register can only be written if the WPEN bit is cleared in the HSMCI Write Protection Mode Register.

- **DTOCYC: Data Timeout Cycle Number**

This field determines the maximum number of Master Clock cycles that the HSMCI waits between two data block transfers. It equals (DTOCYC x Multiplier).

- **DTOMUL: Data Timeout Multiplier**

Value	Name	Description
0	1	DTOCYC
1	16	DTOCYC x 16
2	128	DTOCYC x 128
3	256	DTOCYC x 256
4	1024	DTOCYC x 1024
5	4096	DTOCYC x 4096
6	65536	DTOCYC x 65536
7	1048576	DTOCYC x 1048576

If the data time-out set by DTOCYC and DTOMUL has been exceeded, the Data Time-out Error flag (DTOE) in the HSMCI Status Register (HSMCI_SR) rises.

Downloaded from Arrow.com.

39.14.4 HSMCI SDCard/SDIO Register

Name: HSMCI_SDCR

Address: 0x4000000C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
SDCBUS		–	–	–	–	SDCSEL	

This register can only be written if the WPEN bit is cleared in the HSMCI Write Protection Mode Register.

- **SDCSEL: SDCard/SDIO Slot**

Value	Name	Description
0	SLOTA	Slot A is selected.
1	SLOTB	Reserved
2	SLOTC	Reserved
3	SLOTD	Reserved

- **SDCBUS: SDCard/SDIO Bus Width**

Value	Name	Description
0	1	1 bit
1	–	Reserved
2	4	4 bits
3	8	8 bits

Downloaded from Arrow.com.

39.14.5 HSMCI Argument Register

Name: HSMCI_ARGR

Address: 0x40000010

Access: Read/Write

31	30	29	28	27	26	25	24
ARG							
23	22	21	20	19	18	17	16
ARG							
15	14	13	12	11	10	9	8
ARG							
7	6	5	4	3	2	1	0
ARG							

- **ARG: Command Argument**

Downloaded from Arrow.com.

39.14.6 HSMCI Command Register

Name: HSMCI_CMDR

Address: 0x40000014

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	BOOT_ACK	ATACS	IOSPCMD	
23	22	21	20	19	18	17	16
–	–	TRTYP			TRDIR	TRCMD	
15	14	13	12	11	10	9	8
–	–	–	MAXLAT	OPDCMD	SPCMD		
7	6	5	4	3	2	1	0
RSPTYP		CMDNB					

This register is write-protected while CMDRDY is 0 in HSMCI_SR. If an Interrupt command is sent, this register is only writable by an interrupt response (field SPCMD). This means that the current command execution cannot be interrupted or modified.

- **CMDNB: Command Number**

This is the command index.

- **RSPTYP: Response Type**

Value	Name	Description
0	NORESP	No response
1	48_BIT	48-bit response
2	136_BIT	136-bit response
3	R1B	R1b response type

- **SPCMD: Special Command**

Value	Name	Description
0	STD	Not a special CMD.
1	INIT	Initialization CMD: 74 clock cycles for initialization sequence.
2	SYNC	Synchronized CMD: Wait for the end of the current data block transfer before sending the pending command.
3	CE_ATA	CE-ATA Completion Signal disable Command. The host cancels the ability for the device to return a command completion signal on the command line.
4	IT_CMD	Interrupt command: Corresponds to the Interrupt Mode (CMD40).
5	IT_RESP	Interrupt response: Corresponds to the Interrupt Mode (CMD40).
6	BOR	Boot Operation Request. Start a boot operation mode, the host processor can read boot data from the MMC device directly.
7	EBO	End Boot Operation. This command allows the host processor to terminate the boot operation mode.

Downloaded from Arrow.com.

- **OPDCMD: Open Drain Command**

0 (PUSHPULL): Push pull command.

1 (OPENDRAIN): Open drain command.

- **MAXLAT: Max Latency for Command to Response**

0 (5): 5-cycle max latency.

1 (64): 64-cycle max latency.

- **TRCMD: Transfer Command**

Value	Name	Description
0	NO_DATA	No data transfer
1	START_DATA	Start data transfer
2	STOP_DATA	Stop data transfer
3	–	Reserved

- **TRDIR: Transfer Direction**

0 (WRITE): Write.

1 (READ): Read.

- **TRTYP: Transfer Type**

Value	Name	Description
0	SINGLE	MMC/SD Card Single Block
1	MULTIPLE	MMC/SD Card Multiple Block
2	STREAM	MMC Stream
4	BYTE	SDIO Byte
5	BLOCK	SDIO Block

- **IOSPCMD: SDIO Special Command**

Value	Name	Description
0	STD	Not an SDIO Special Command
1	SUSPEND	SDIO Suspend Command
2	RESUME	SDIO Resume Command

- **ATACS: ATA with Command Completion Signal**

0 (NORMAL): Normal operation mode.

1 (COMPLETION): This bit indicates that a completion signal is expected within a programmed amount of time (HSMCI_CSTOR).

- **BOOT_ACK: Boot Operation Acknowledge**

The master can choose to receive the boot acknowledge from the slave when a Boot Request command is issued. When set to one this field indicates that a Boot acknowledge is expected within a programmable amount of time defined with DTOMUL and DTOCYC fields located in the HSMCI_DTOR. If the acknowledge pattern is not received then an acknowledge timeout error is raised. If the acknowledge pattern is corrupted then an acknowledge pattern error is set.

Downloaded from Arrow.com.

39.14.7 HSMCI Block Register

Name: HSMCI_BLKR

Address: 0x40000018

Access: Read/Write

31	30	29	28	27	26	25	24
BLKLEN							
23	22	21	20	19	18	17	16
BLKLEN							
15	14	13	12	11	10	9	8
BCNT							
7	6	5	4	3	2	1	0
BCNT							

- **BCNT: MMC/SDIO Block Count - SDIO Byte Count**

This field determines the number of data byte(s) or block(s) to transfer.

The transfer data type and the authorized values for BCNT field are determined by the TRTYP field in the HSMCI Command Register (HSMCI_CMDR).

When TRTYP = 1 (MMC/SDCARD Multiple Block), BCNT can be programmed from 1 to 65535, 0 corresponds to an infinite block transfer.

When TRTYP = 4 (SDIO Byte), BCNT can be programmed from 1 to 511, 0 corresponds to 512-byte transfer. Values in range 512 to 65536 are forbidden.

When TRTYP = 5 (SDIO Block), BCNT can be programmed from 1 to 511, 0 corresponds to an infinite block transfer. Values in range 512 to 65536 are forbidden.

Warning: In SDIO Byte and Block modes (TRTYP = 4 or 5), writing the 7 last bits of BCNT field with a value which differs from 0 is forbidden and may lead to unpredictable results.

- **BLKLEN: Data Block Length**

This field determines the size of the data block.

Bits 16 and 17 must be configured to 0 if FBYTE is disabled.

Note: In SDIO Byte mode, BLKLEN field is not used.

Downloaded from Arrow.com.

39.14.8 HSMCI Completion Signal Timeout Register

Name: HSMCI_CSTOR

Address: 0x4000001C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	CSTOMUL			CSTOCYC			

This register can only be written if the WPEN bit is cleared in the HSMCI Write Protection Mode Register.

- **CSTOCYC: Completion Signal Timeout Cycle Number**

This field determines the maximum number of Master Clock cycles that the HSMCI waits between two data block transfers. Its value is calculated by (CSTOCYC x Multiplier).

- **CSTOMUL: Completion Signal Timeout Multiplier**

This field determines the maximum number of Master Clock cycles that the HSMCI waits between two data block transfers. Its value is calculated by (CSTOCYC x Multiplier).

These fields determine the maximum number of Master Clock cycles that the HSMCI waits between the end of the data transfer and the assertion of the completion signal. The data transfer comprises data phase and the optional busy phase. If a non-DATA ATA command is issued, the HSMCI starts waiting immediately after the end of the response until the completion signal.

Multiplier is defined by CSTOMUL as shown in the following table:

Value	Name	Description
0	1	CSTOCYC x 1
1	16	CSTOCYC x 16
2	128	CSTOCYC x 128
3	256	CSTOCYC x 256
4	1024	CSTOCYC x 1024
5	4096	CSTOCYC x 4096
6	65536	CSTOCYC x 65536
7	1048576	CSTOCYC x 1048576

If the data time-out set by CSTOCYC and CSTOMUL has been exceeded, the Completion Signal Time-out Error flag (CSTOE) in the HSMCI Status Register (HSMCI_SR) rises.

Downloaded from Arrow.com.

39.14.9 HSMCI Response Register

Name: HSMCI_RSPR

Address: 0x40000020

Access: Read-only

31	30	29	28	27	26	25	24
RSP							
23	22	21	20	19	18	17	16
RSP							
15	14	13	12	11	10	9	8
RSP							
7	6	5	4	3	2	1	0
RSP							

- **RSP: Response**

Note: 1. The response register can be read by N accesses at the same HSMCI_RSPR or at consecutive addresses (0x20 to 0x2C). N depends on the size of the response.

Downloaded from Arrow.com.

39.14.10 HSMCI Receive Data Register

Name: HSMCI_RDR

Address: 0x40000030

Access: Read-only

31	30	29	28	27	26	25	24
DATA							
23	22	21	20	19	18	17	16
DATA							
15	14	13	12	11	10	9	8
DATA							
7	6	5	4	3	2	1	0
DATA							

- **DATA: Data to Read**

Downloaded from Arrow.com.

39.14.11 HSMCI Transmit Data Register

Name: HSMCI_TDR

Address: 0x40000034

Access: Write-only

31	30	29	28	27	26	25	24
DATA							
23	22	21	20	19	18	17	16
DATA							
15	14	13	12	11	10	9	8
DATA							
7	6	5	4	3	2	1	0
DATA							

- **DATA: Data to Write**

Downloaded from Arrow.com.

39.14.12 HSMCI Status Register

Name: HSMCI_SR

Address: 0x40000040

Access: Read-only

31	30	29	28	27	26	25	24
UNRE	OVRE	ACKRCVE	ACKRCV	XFRDONE	FIFOEMPTY	–	BLKOVRE
23	22	21	20	19	18	17	16
CSTOE	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
–	–	CSRCV	SDIOWAIT	–	–	–	SDIOIRQA
7	6	5	4	3	2	1	0
–	–	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

- **CMDRDY: Command Ready (cleared by writing in HSMCI_CMDR)**

0: A command is in progress.

1: The last command has been sent.

- **RXRDY: Receiver Ready (cleared by reading HSMCI_RDR)**

0: Data has not yet been received since the last read of HSMCI_RDR.

1: Data has been received since the last read of HSMCI_RDR.

- **TXRDY: Transmit Ready (cleared by writing in HSMCI_TDR)**

0: The last data written in HSMCI_TDR has not yet been transferred in the Shift Register.

1: The last data written in HSMCI_TDR has been transferred in the Shift Register.

- **BLKE: Data Block Ended (cleared on read)**

This flag must be used only for Write Operations.

0: A data block transfer is not yet finished.

1: A data block transfer has ended, including the CRC16 Status transmission. The flag is set for each transmitted CRC Status.

Refer to the MMC or SD Specification for more details concerning the CRC Status.

- **DTIP: Data Transfer in Progress (cleared at the end of CRC16 calculation)**

0: No data transfer in progress.

1: The current data transfer is still in progress, including CRC16 calculation.

Downloaded from Arrow.com.

- **NOTBUSY: HSMCI Not Busy**

A block write operation uses a simple busy signalling of the write operation duration on the data (DAT0) line: during a data transfer block, if the card does not have a free data receive buffer, the card indicates this condition by pulling down the data line (DAT0) to LOW. The card stops pulling down the data line as soon as at least one receive buffer for the defined data transfer block length becomes free.

Refer to the MMC or SD Specification for more details concerning the busy behavior.

For all the read operations, the NOTBUSY flag is cleared at the end of the host command.
For the Infinite Read Multiple Blocks, the NOTBUSY flag is set at the end of the STOP_TRANSMISSION host command (CMD12).

For the Single Block Reads, the NOTBUSY flag is set at the end of the data read block.

For the Multiple Block Reads with predefined block count, the NOTBUSY flag is set at the end of the last received data block.

The NOTBUSY flag allows to deal with these different states.

0: The HSMCI is not ready for new data transfer. Cleared at the end of the card response.

1: The HSMCI is ready for new data transfer. Set when the busy state on the data line has ended. This corresponds to a free internal data receive buffer of the card.

- **SDIOIRQA: SDIO Interrupt for Slot A (cleared on read)**

0: No interrupt detected on SDIO Slot A.

1: An SDIO Interrupt on Slot A occurred.

- **SDIOWAIT: SDIO Read Wait Operation Status**

0: Normal Bus operation.

1: The data bus has entered IO wait state.

- **CSRCV: CE-ATA Completion Signal Received (cleared on read)**

0: No completion signal received since last status read operation.

1: The device has issued a command completion signal on the command line.

- **RINDE: Response Index Error (cleared by writing in HSMCI_CMDR)**

0: No error.

1: A mismatch is detected between the command index sent and the response index received.

- **RDIRE: Response Direction Error (cleared by writing in HSMCI_CMDR)**

0: No error.

1: The direction bit from card to host in the response has not been detected.

- **RCRCE: Response CRC Error (cleared by writing in HSMCI_CMDR)**

0: No error.

1: A CRC7 error has been detected in the response.

- **RENDE: Response End Bit Error (cleared by writing in HSMCI_CMDR)**

0: No error.

1: The end bit of the response has not been detected.

Downloaded from Arrow.com.

- **RTOE: Response Time-out Error (cleared by writing in HSMCI_CMDR)**

0: No error.

1: The response time-out set by MAXLAT in the HSMCI_CMDR has been exceeded.

- **DCRCE: Data CRC Error (cleared on read)**

0: No error.

1: A CRC16 error has been detected in the last data block.

- **DTOE: Data Time-out Error (cleared on read)**

0: No error.

1: The data time-out set by DTOCYC and DTOMUL in HSMCI_DTOR has been exceeded.

- **CSTOE: Completion Signal Time-out Error (cleared on read)**

0: No error.

1: The completion signal time-out set by CSTOCYC and CSTOMUL in HSMCI_CSTOR has been exceeded.

- **BLKOVRE: DMA Block Overrun Error (cleared on read)**

0: No error.

1: A new block of data is received and the DMA controller has not started to move the current pending block, a block overrun is raised.

- **FIFOEMPTY: FIFO empty flag**

0: FIFO contains at least one byte.

1: FIFO is empty.

- **XFRDONE: Transfer Done flag**

0: A transfer is in progress.

1: Command Register is ready to operate and the data bus is in the idle state.

- **ACKRCV: Boot Operation Acknowledge Received (cleared on read)**

0: No Boot acknowledge received since the last read of the HSMCI_SR.

1: A Boot acknowledge signal has been received since the last read of HSMCI_SR.

- **ACKRCVE: Boot Operation Acknowledge Error (cleared on read)**

0: No boot operation error since the last read of HSMCI_SR

1: Corrupted Boot Acknowledge signal received since the last read of HSMCI_SR.

- **OVRE: Overrun (if FERRCTRL = 1, cleared by writing in HSMCI_CMDR or cleared on read if FERRCTRL = 0)**

0: No error.

1: At least one 8-bit received data has been lost (not read).

If FERRCTRL = 1 in HSMCI_CFG, OVRE is cleared on read.

If FERRCTRL = 0 in HSMCI_CFG, OVRE is cleared by writing HSMCI_CMDR.

Downloaded from Arrow.com.

- **UNRE: Underrun (if FERRCTRL = 1, cleared by writing in HSMCI_CMDR or cleared on read if FERRCTRL = 0)**

0: No error.

1: At least one 8-bit data has been sent without valid information (not written).

If FERRCTRL = 1 in HSMCI_CFG, OVRE is cleared on read.

If FERRCTRL = 0 in HSMCI_CFG, OVRE is cleared by writing HSMCI_CMDR.

Downloaded from Arrow.com.

39.14.13 HSMCI Interrupt Enable Register

Name: HSMCI_IER

Address: 0x40000044

Access: Write-only

31	30	29	28	27	26	25	24
UNRE	OVRE	ACKRCVE	ACKRCV	XFRDONE	FIFOEMPTY	–	BLKOVRE
23	22	21	20	19	18	17	16
CSTOE	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
–	–	CSRCV	SDIOWAIT	–	–	–	SDIOIRQA
7	6	5	4	3	2	1	0
–	–	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

- **CMDRDY: Command Ready Interrupt Enable**
- **RXRDY: Receiver Ready Interrupt Enable**
- **TXRDY: Transmit Ready Interrupt Enable**
- **BLKE: Data Block Ended Interrupt Enable**
- **DTIP: Data Transfer in Progress Interrupt Enable**
- **NOTBUSY: Data Not Busy Interrupt Enable**
- **SDIOIRQA: SDIO Interrupt for Slot A Interrupt Enable**
- **SDIOWAIT: SDIO Read Wait Operation Status Interrupt Enable**
- **CSRCV: Completion Signal Received Interrupt Enable**
- **RINDE: Response Index Error Interrupt Enable**
- **RDIRE: Response Direction Error Interrupt Enable**
- **RCRCE: Response CRC Error Interrupt Enable**
- **RENDE: Response End Bit Error Interrupt Enable**
- **RTOE: Response Time-out Error Interrupt Enable**
- **DCRCE: Data CRC Error Interrupt Enable**
- **DTOE: Data Time-out Error Interrupt Enable**
- **CSTOE: Completion Signal Timeout Error Interrupt Enable**

Downloaded from Arrow.com.

- **BLKOVRE: DMA Block Overrun Error Interrupt Enable**
- **FIFOEMPTY: FIFO empty Interrupt enable**
- **XFRDONE: Transfer Done Interrupt enable**
- **ACKRCV: Boot Acknowledge Interrupt Enable**
- **ACKRCVE: Boot Acknowledge Error Interrupt Enable**
- **OVRE: Overrun Interrupt Enable**
- **UNRE: Underrun Interrupt Enable**

Downloaded from Arrow.com.

39.14.14 HSMCI Interrupt Disable Register

Name: HSMCI_IDR

Address: 0x40000048

Access: Write-only

31	30	29	28	27	26	25	24
UNRE	OVRE	ACKRCVE	ACKRCV	XFRDONE	FIFOEMPTY	–	BLKOVRE
23	22	21	20	19	18	17	16
CSTOE	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
–	–	CSRCV	SDIOWAIT	–	–	–	SDIOIRQA
7	6	5	4	3	2	1	0
–	–	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

- **CMDRDY: Command Ready Interrupt Disable**
- **RXRDY: Receiver Ready Interrupt Disable**
- **TXRDY: Transmit Ready Interrupt Disable**
- **BLKE: Data Block Ended Interrupt Disable**
- **DTIP: Data Transfer in Progress Interrupt Disable**
- **NOTBUSY: Data Not Busy Interrupt Disable**
- **SDIOIRQA: SDIO Interrupt for Slot A Interrupt Disable**
- **SDIOWAIT: SDIO Read Wait Operation Status Interrupt Disable**
- **CSRCV: Completion Signal received interrupt Disable**
- **RINDE: Response Index Error Interrupt Disable**
- **RDIRE: Response Direction Error Interrupt Disable**
- **RCRCE: Response CRC Error Interrupt Disable**
- **RENDE: Response End Bit Error Interrupt Disable**
- **RTOE: Response Time-out Error Interrupt Disable**
- **DCRCE: Data CRC Error Interrupt Disable**
- **DTOE: Data Time-out Error Interrupt Disable**
- **CSTOE: Completion Signal Time out Error Interrupt Disable**

Downloaded from Arrow.com.

- **BLKOVRE: DMA Block Overrun Error Interrupt Disable**
- **FIFOEMPTY: FIFO empty Interrupt Disable**
- **XFRDONE: Transfer Done Interrupt Disable**
- **ACKRCV: Boot Acknowledge Interrupt Disable**
- **ACKRCVE: Boot Acknowledge Error Interrupt Disable**
- **OVRE: Overrun Interrupt Disable**
- **UNRE: Underrun Interrupt Disable**

Downloaded from Arrow.com.

39.14.15 HSMCI Interrupt Mask Register

Name: HSMCI_IMR

Address: 0x4000004C

Access: Read-only

31	30	29	28	27	26	25	24
UNRE	OVRE	ACKRCVE	ACKRCV	XFRDONE	FIFOEMPTY	–	BLKOVRE
23	22	21	20	19	18	17	16
CSTOE	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
–	–	CSRCV	SDIOWAIT	–	–	–	SDIOIRQA
7	6	5	4	3	2	1	0
–	–	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

- **CMDRDY: Command Ready Interrupt Mask**
- **RXRDY: Receiver Ready Interrupt Mask**
- **TXRDY: Transmit Ready Interrupt Mask**
- **BLKE: Data Block Ended Interrupt Mask**
- **DTIP: Data Transfer in Progress Interrupt Mask**
- **NOTBUSY: Data Not Busy Interrupt Mask**
- **SDIOIRQA: SDIO Interrupt for Slot A Interrupt Mask**
- **SDIOWAIT: SDIO Read Wait Operation Status Interrupt Mask**
- **CSRCV: Completion Signal Received Interrupt Mask**
- **RINDE: Response Index Error Interrupt Mask**
- **RDIRE: Response Direction Error Interrupt Mask**
- **RCRCE: Response CRC Error Interrupt Mask**
- **RENDE: Response End Bit Error Interrupt Mask**
- **RTOE: Response Time-out Error Interrupt Mask**
- **DCRCE: Data CRC Error Interrupt Mask**
- **DTOE: Data Time-out Error Interrupt Mask**
- **CSTOE: Completion Signal Time-out Error Interrupt Mask**

Downloaded from Arrow.com.

- **BLKOVRE: DMA Block Overrun Error Interrupt Mask**
- **FIFOEMPTY: FIFO Empty Interrupt Mask**
- **XFRDONE: Transfer Done Interrupt Mask**
- **ACKRCV: Boot Operation Acknowledge Received Interrupt Mask**
- **ACKRCVE: Boot Operation Acknowledge Error Interrupt Mask**
- **OVRE: Overrun Interrupt Mask**
- **UNRE: Underrun Interrupt Mask**

Downloaded from Arrow.com.

39.14.16 HSMCI DMA Configuration Register

Name: HSMCI_DMA

Address: 0x40000050

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	DMAEN
7	6	5	4	3	2	1	0
–	CHKSIZE			–	–	–	

This register can only be written if the WPEN bit is cleared in the HSMCI Write Protection Mode Register.

- **CHKSIZE: DMA Channel Read and Write Chunk Size**

The CHKSIZE field indicates the number of data available when the DMA chunk transfer request is asserted.

Value	Name	Description
0	1	1 data available
1	2	2 data available
2	4	4 data available
3	8	8 data available
4	16	16 data available

- **DMAEN: DMA Hardware Handshaking Enable**

0: DMA interface is disabled.

1: DMA Interface is enabled.

Note: To avoid unpredictable behavior, DMA hardware handshaking must be disabled when CPU transfers are performed.

Downloaded from Arrow.com.

39.14.17 HSMCI Configuration Register

Name: HSMCI_CFG

Address: 0x40000054

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	LSYNC	–	–	–	HSMODE
7	6	5	4	3	2	1	0
–	–	–	FERRCTRL	–	–	–	FIFOMODE

This register can only be written if the WPEN bit is cleared in the HSMCI Write Protection Mode Register.

- **FIFOMODE: HSMCI Internal FIFO control mode**

0: A write transfer starts when a sufficient amount of data is written into the FIFO.

When the block length is greater than or equal to 3/4 of the HSMCI internal FIFO size, then the write transfer starts as soon as half the FIFO is filled. When the block length is greater than or equal to half the internal FIFO size, then the write transfer starts as soon as one quarter of the FIFO is filled. In other cases, the transfer starts as soon as the total amount of data is written in the internal FIFO.

1: A write transfer starts as soon as one data is written into the FIFO.

- **FERRCTRL: Flow Error flag reset control mode**

0: When an underflow/overflow condition flag is set, a new Write/Read command is needed to reset the flag.

1: When an underflow/overflow condition flag is set, a read status resets the flag.

- **HSMODE: High Speed Mode**

0: Default bus timing mode.

1: If set to one, the host controller outputs command line and data lines on the rising edge of the card clock. The Host driver shall check the high speed support in the card registers.

- **LSYNC: Synchronize on the last block**

0: The pending command is sent at the end of the current data block.

1: The pending command is sent at the end of the block transfer when the transfer length is not infinite (block count shall be different from zero).

Downloaded from Arrow.com.

39.14.18 HSMCI Write Protection Mode Register

Name: HSMCI_WPMR

Address: 0x400000E4

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							
23	22	21	20	19	18	17	16
WPKEY							
15	14	13	12	11	10	9	8
WPKEY							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

- **WPEN: Write Protect Enable**

0: Disables the Write Protection if WPKEY corresponds to 0x4D4349 (“MCI” in ASCII).

1: Enables the Write Protection if WPKEY corresponds to 0x4D4349 (“MCI” in ASCII).

See Section 39.13 “Register Write Protection” for the list of registers that can be write-protected.

- **WPKEY: Write Protect Key**

Value	Name	Description
0x4D4349	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

Downloaded from Arrow.com.

39.14.19 HSMCI Write Protection Status Register

Name: HSMCI_WPSR

Address: 0x400000E8

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
WPVSRC							
15	14	13	12	11	10	9	8
WPVSRC							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

- **WPVS: Write Protection Violation Status**

0: No write protection violation has occurred since the last read of the HSMCI_WPSR.

1: A write protection violation has occurred since the last read of the HSMCI_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protection Violation Source**

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

Downloaded from Arrow.com.

39.14.20 HSMCI FIFOx Memory Aperture

Name: HSMCI_FIFOx [x=0..255]

Address: 0x40000200

Access: Read/Write

31	30	29	28	27	26	25	24
DATA							
23	22	21	20	19	18	17	16
DATA							
15	14	13	12	11	10	9	8
DATA							
7	6	5	4	3	2	1	0
DATA							

- **DATA: Data to Read or Data to Write**

Downloaded from Arrow.com.

40. Serial Peripheral Interface (SPI)

40.1 Description

The Serial Peripheral Interface (SPI) circuit is a synchronous serial data link that provides communication with external devices in Master or Slave mode. It also enables communication between processors if an external processor is connected to the system.

The Serial Peripheral Interface is essentially a Shift register that serially transmits data bits to other SPIs. During a data transfer, one SPI system acts as the “master”' which controls the data flow, while the other devices act as “slaves'' which have data shifted into and out by the master. Different CPUs can take turn being masters (multiple master protocol, contrary to single master protocol where one CPU is always the master while all of the others are always slaves). One master can simultaneously shift data into multiple slaves. However, only one slave can drive its output to write data back to the master at any given time.

A slave device is selected when the master asserts its NSS signal. If multiple slave devices exist, the master generates a separate slave select signal for each slave (NPCS).

The SPI system consists of two data lines and two control lines:

- Master Out Slave In (MOSI)—This data line supplies the output data from the master shifted into the input(s) of the slave(s).
- Master In Slave Out (MISO)—This data line supplies the output data from a slave to the input of the master. There may be no more than one slave transmitting data during any particular transfer.
- Serial Clock (SPCK)—This control line is driven by the master and regulates the flow of the data bits. The master can transmit data at a variety of baud rates; there is one SPCK pulse for each bit that is transmitted.
- Slave Select (NSS)—This control line allows slaves to be turned on and off by hardware.

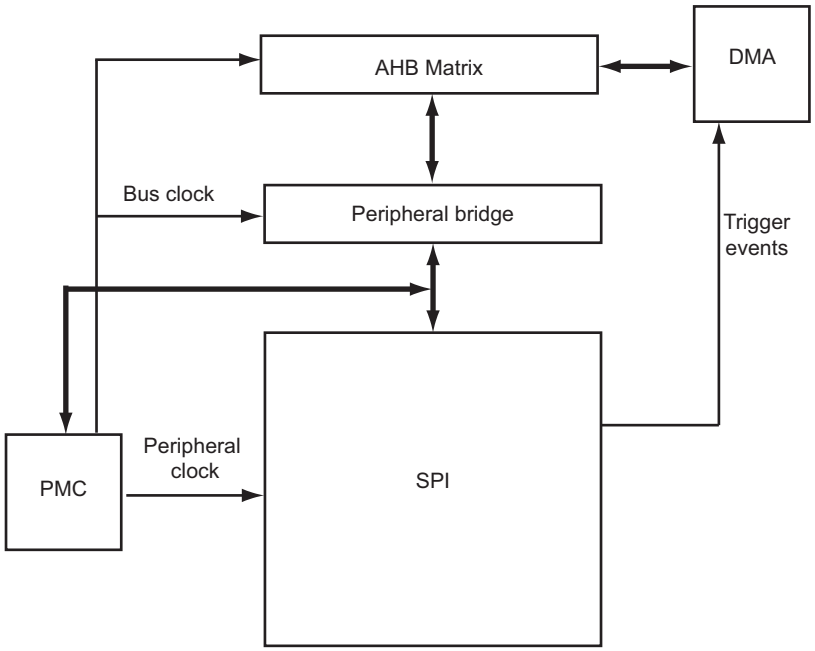
40.2 Embedded Characteristics

- Master or Slave Serial Peripheral Bus Interface
  - 8-bit to 16-bit programmable data length per chip select
  - Programmable phase and polarity per chip select
  - Programmable transfer delay between consecutive transfers and delay before SPI clock per chip select
  - Programmable delay between chip selects
  - Selectable mode fault detection
- Master Mode can drive SPCK up to Peripheral Clock
- Master Mode Bit Rate can be Independent of the Processor/Peripheral Clock
- Slave mode operates on SPCK, asynchronously with core and bus clock
- Four chip selects with external decoder support allow communication with up to 15 peripherals
- Communication with Serial External Devices Supported
  - Serial memories, such as DataFlash and 3-wire EEPROMs
  - Serial peripherals, such as ADCs, DACs, LCD controllers, CAN controllers and sensors
  - External coprocessors
- Connection to DMA Channel Capabilities, Optimizing Data Transfers
  - One channel for the receiver
  - One channel for the transmitter
- Register Write Protection

Downloaded from Arrow.com.

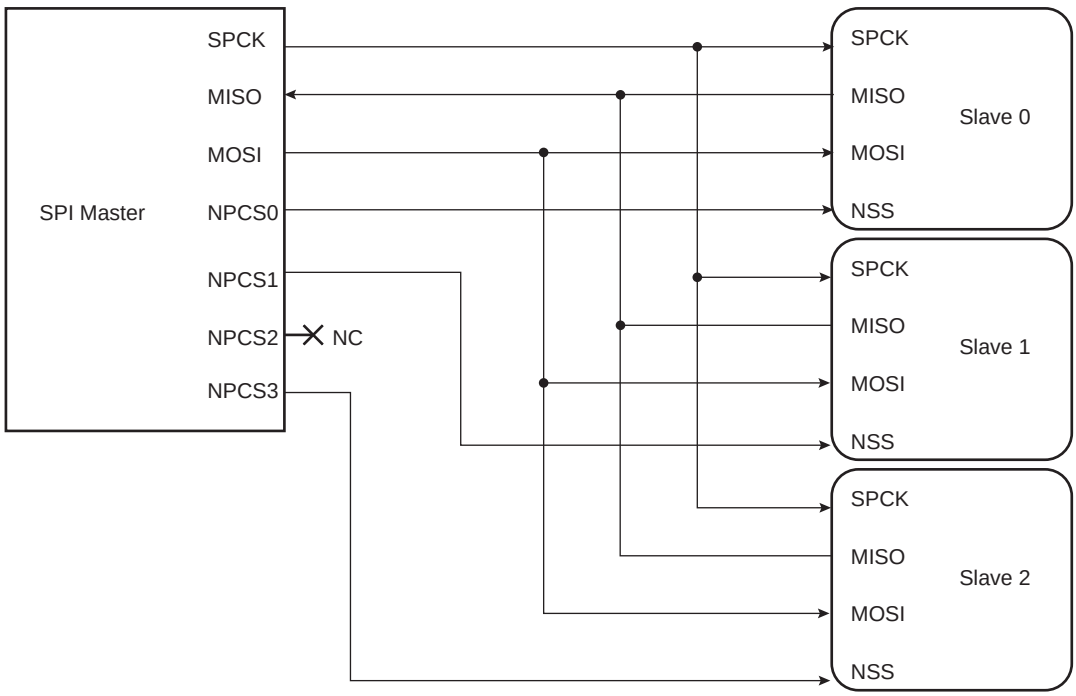
40.3 Block Diagram

Figure 40-1. Block Diagram



40.4 Application Block Diagram

Figure 40-2. Application Block Diagram: Single Master/Multiple Slave Implementation



Downloaded from Arrow.com.

40.5 Signal Description

Table 40-1. Signal Description

Pin Name	Pin Description	Type	
		Master	Slave
MISO	Master In Slave Out	Input	Output
MOSI	Master Out Slave In	Output	Input
SPCK	Serial Clock	Output	Input
NPCS1–NPCS3	Peripheral Chip Selects	Output	Unused
NPCS0/NSS	Peripheral Chip Select/Slave Select	Output	Input

40.6 Product Dependencies

40.6.1 I/O Lines

The pins used for interfacing the compliant external devices can be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the SPI pins to their peripheral functions.

Table 40-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
SPI0	SPI0_MISO	PD20	B
SPI0	SPI0_MOSI	PD21	B
SPI0	SPI0_NPCS0	PB2	D
SPI0	SPI0_NPCS1	PA31	A
SPI0	SPI0_NPCS1	PD25	B
SPI0	SPI0_NPCS2	PD12	C
SPI0	SPI0_NPCS3	PD27	B
SPI0	SPI0_SPCK	PD22	B
SPI1	SPI1_MISO	PC26	C
SPI1	SPI1_MOSI	PC27	C
SPI1	SPI1_NPCS0	PC25	C
SPI1	SPI1_NPCS1	PC28	C
SPI1	SPI1_NPCS1	PD0	C
SPI1	SPI1_NPCS2	PC29	C
SPI1	SPI1_NPCS2	PD1	C
SPI1	SPI1_NPCS3	PC30	C
SPI1	SPI1_NPCS3	PD2	C
SPI1	SPI1_SPCK	PC24	C

40.6.2 Power Management

The SPI can be clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the SPI clock.

Downloaded from Arrow.com.

40.6.3 Interrupt

The SPI interface has an interrupt line connected to the interrupt controller. Handling the SPI interrupt requires programming the interrupt controller before configuring the SPI.

Table 40-3. Peripheral IDs

Instance	ID
SPI0	21
SPI1	42

40.6.4 Direct Memory Access Controller (DMAC)

The SPI interface can be used in conjunction with the DMAC in order to reduce processor overhead. For a full description of the DMAC, refer to Section 35. “DMA Controller (XDMAC)”.

40.7 Functional Description

40.7.1 Modes of Operation

The SPI operates in Master mode or in Slave mode.

- The SPI operates in Master mode by setting the MSTR bit in the SPI Mode Register (SPI_MR):
  - Pins NPCS0 to NPCS3 are all configured as outputs
  - The SPCK pin is driven
  - The MISO line is wired on the receiver input
  - The MOSI line is driven as an output by the transmitter.
- The SPI operates in Slave mode if the MSTR bit in the SPI_MR is written to 0:
  - The MISO line is driven by the transmitter output
  - The MOSI line is wired on the receiver input
  - The SPCK pin is driven by the transmitter to synchronize the receiver.
  - The NPCS0 pin becomes an input, and is used as a slave select signal (NSS)
  - NPCS1 to NPCS3 are not driven and can be used for other purposes.

The data transfers are identically programmable for both modes of operation. The baud rate generator is activated only in Master mode.

40.7.2 Data Transfer

Four combinations of polarity and phase are available for data transfers. The clock polarity is programmed with the CPOL bit in the SPI chip select registers (SPI_CSRx). The clock phase is programmed with the NCPHA bit. These two parameters determine the edges of the clock signal on which data is driven and sampled. Each of the two parameters has two possible states, resulting in four possible combinations that are incompatible with one another. Consequently, a master/slave pair must use the same parameter pair values to communicate. If multiple slaves are connected and require different configurations, the master must reconfigure itself each time it needs to communicate with a different slave.

Table 40-4 shows the four modes and corresponding parameter settings.

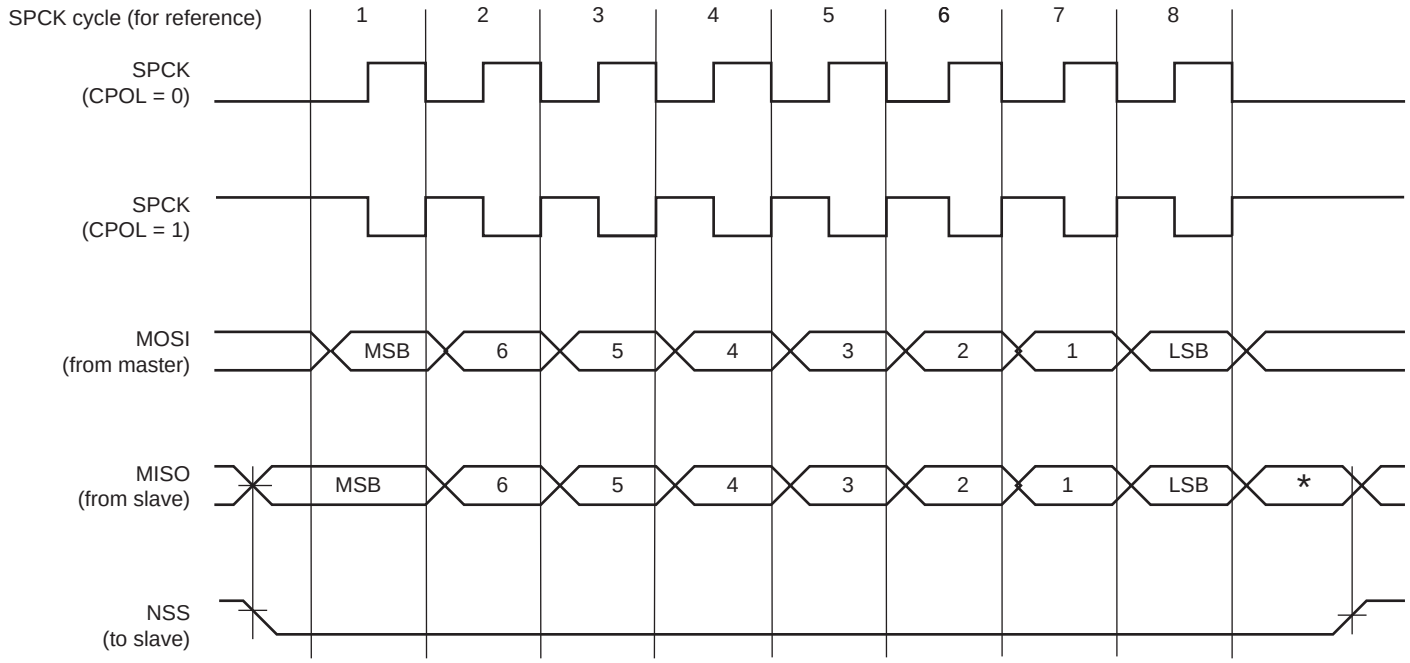
Downloaded from Arrow.com.

Table 40-4. SPI Bus Protocol Modes

SPI Mode	CPOL	NCPHA	Shift SPCK Edge	Capture SPCK Edge	SPCK Inactive Level
0	0	1	Falling	Rising	Low
1	0	0	Rising	Falling	Low
2	1	1	Rising	Falling	High
3	1	0	Falling	Rising	High

Figure 40-3 and Figure 40-4 show examples of data transfers.

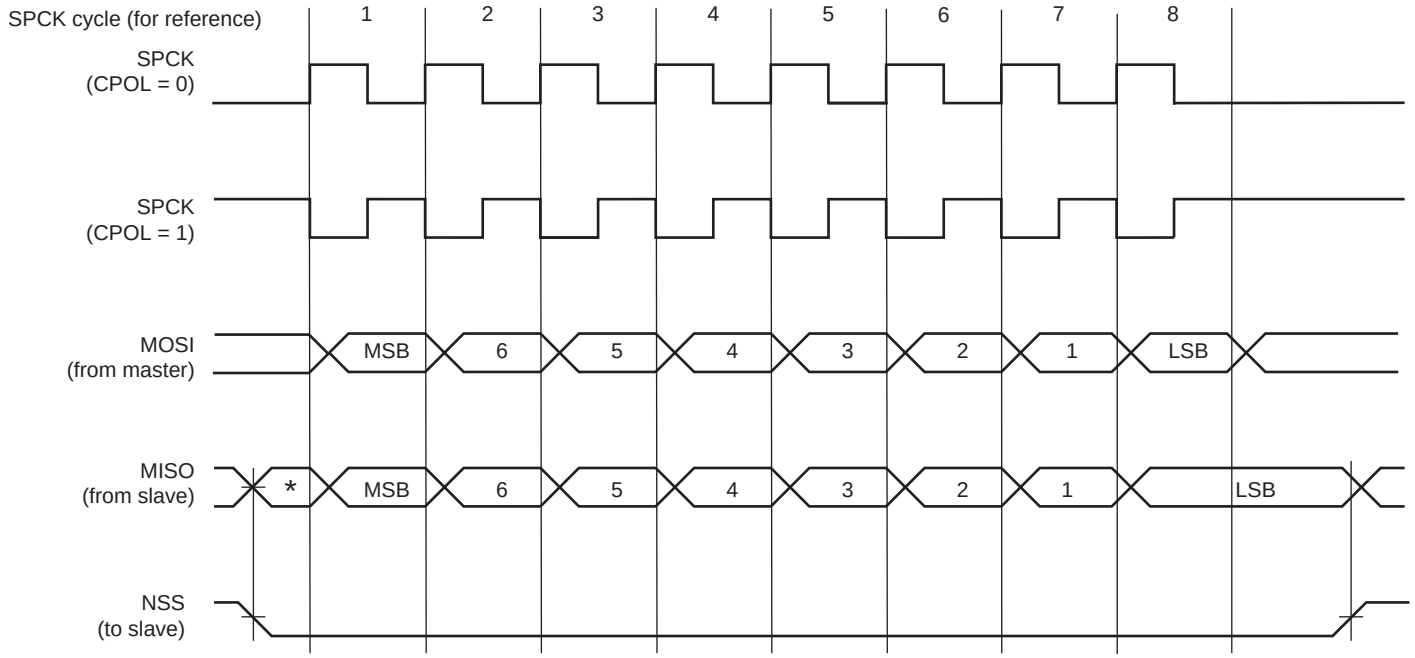
Figure 40-3. SPI Transfer Format (NCPHA = 1, 8 bits per transfer)



* Not defined.

Downloaded from Arrow.com.

Figure 40-4. SPI Transfer Format (NCPHA = 0, 8 bits per transfer)



* Not defined.

40.7.3 Master Mode Operations

When configured in Master mode, the SPI operates on the clock generated by the internal programmable baud rate generator. It fully controls the data transfers to and from the slave(s) connected to the SPI bus. The SPI drives the chip select line to the slave and the serial clock signal (SPCK).

The SPI features two holding registers, the Transmit Data Register (SPI_TDR) and the Receive Data Register (SPI_RDR), and a single shift register. The holding registers maintain the data flow at a constant rate.

After enabling the SPI, a data transfer starts when the processor writes to the SPI_TDR. The written data is immediately transferred in the Shift register and the transfer on the SPI bus starts. While the data in the Shift register is shifted on the MOSI line, the MISO line is sampled and shifted in the Shift register. Data cannot be loaded in the SPI_RDR without transmitting data. If there is no data to transmit, dummy data can be used (SPI_TDR filled with ones). When the SPI_MR.WDRBT bit is set, new data cannot be transmitted if the SPI_RDR has not been read. If Receiving mode is not required, for example when communicating with a slave receiver only (such as an LCD), the receive status flags in the SPI Status register (SPI_SR) can be discarded.

Before writing the SPI_TDR, the PCS field in the SPI_MR must be set in order to select a slave.

If new data is written in the SPI_TDR during the transfer, it is kept in the SPI_TDR until the current transfer is completed. Then, the received data is transferred from the Shift register to the SPI_RDR, the data in the SPI_TDR is loaded in the Shift register and a new transfer starts.

As soon as the SPI_TDR is written, the Transmit Data Register Empty (TDRE) flag in the SPI_SR is cleared. When the data written in the SPI_TDR is loaded into the Shift register, the TDRE flag in the SPI_SR is set. The TDRE bit is used to trigger the Transmit DMA channel.

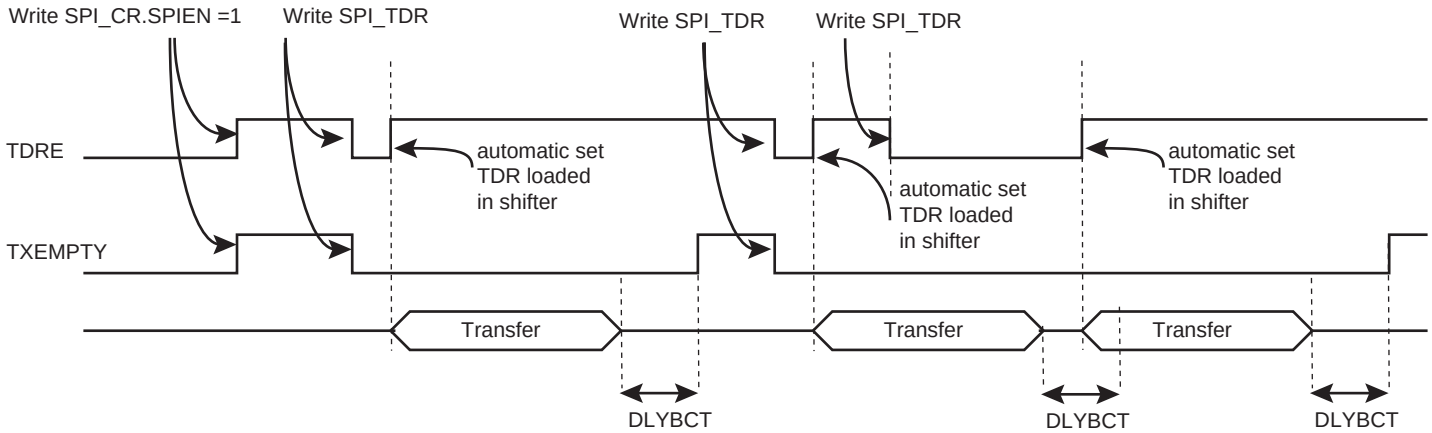
See Figure 40-5.

The end of transfer is indicated by the TXEMPTY flag in the SPI_SR. If a transfer delay (DLYBCT) is greater than 0 for the last transfer, TXEMPTY is set after the completion of this delay. The peripheral clock can be switched off at this time.

Note: When the SPI is enabled, the TDRE and TXEMPTY flags are set.

Downloaded from Arrow.com.

Figure 40-5. TDRE and TXEMPTY flag behavior



The transfer of received data from the Shift register to the SPI_RDR is indicated by the Receive Data Register Full (RDRF) bit in the SPI_SR. When the received data is read, the RDRF bit is cleared.

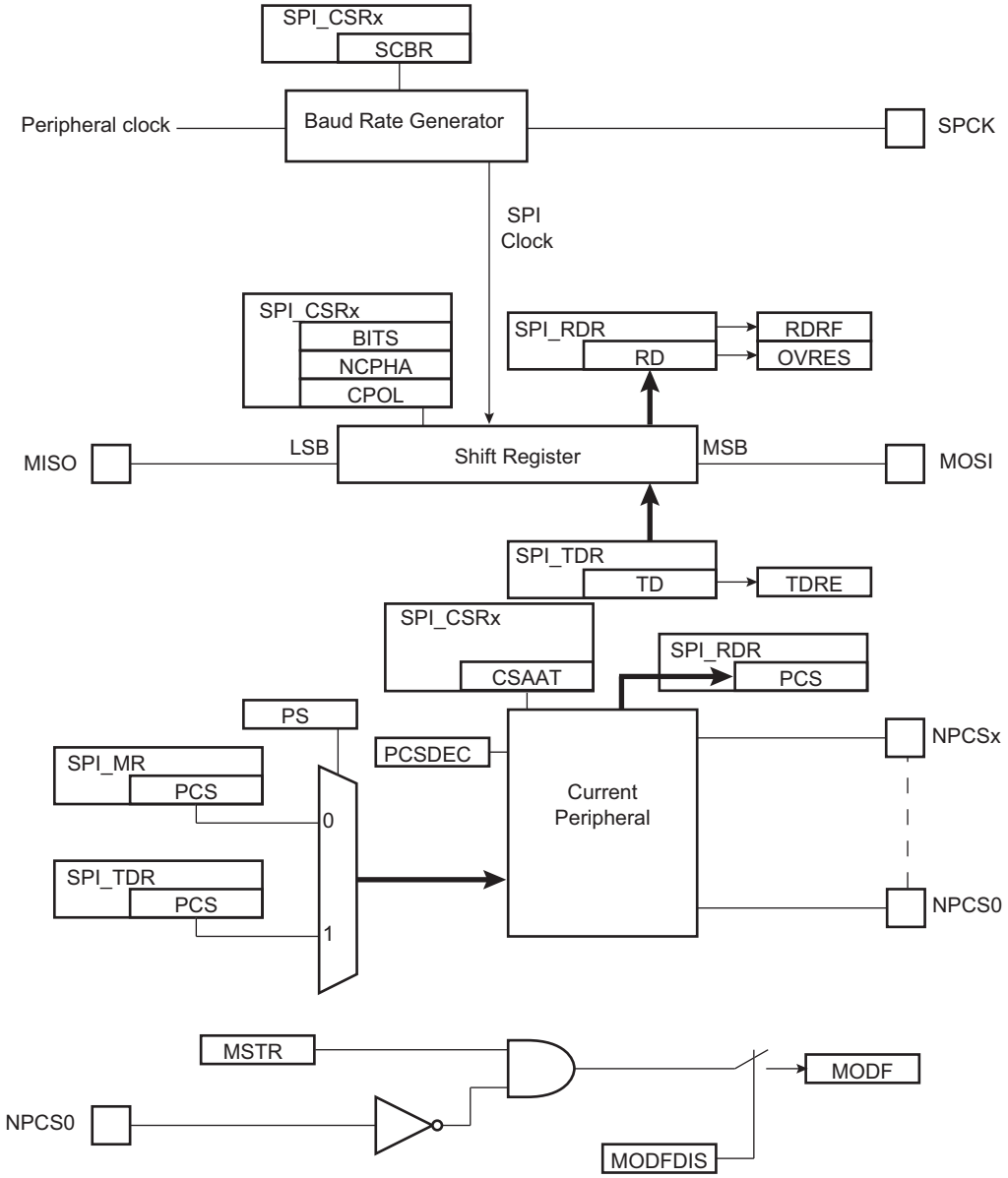
If the SPI_RDR has not been read before new data is received, the Overrun Error (OVRES) bit in the SPI_SR is set. As long as this flag is set, data is loaded in the SPI_RDR. The user has to read the SPI_SR to clear the OVRES bit.

Figure 40-6 shows a block diagram of the SPI when operating in Master mode. Figure 40-7 shows a flow chart describing how transfers are handled.

Downloaded from Arrow.com.

40.7.3.1 Master Mode Block Diagram

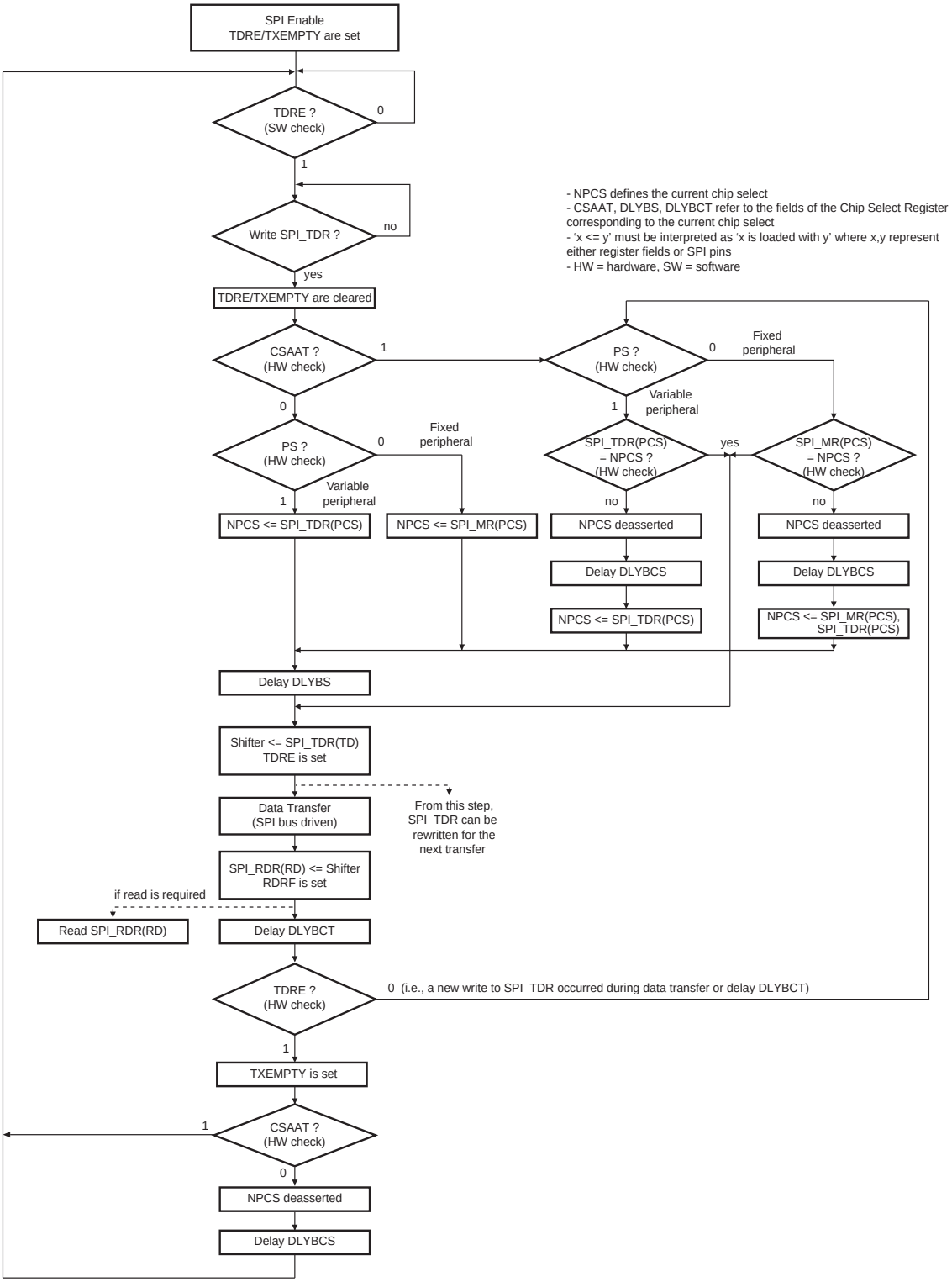
Figure 40-6. Master Mode Block Diagram



Downloaded from Arrow.com.

40.7.3.2 Master Mode Flow Diagram

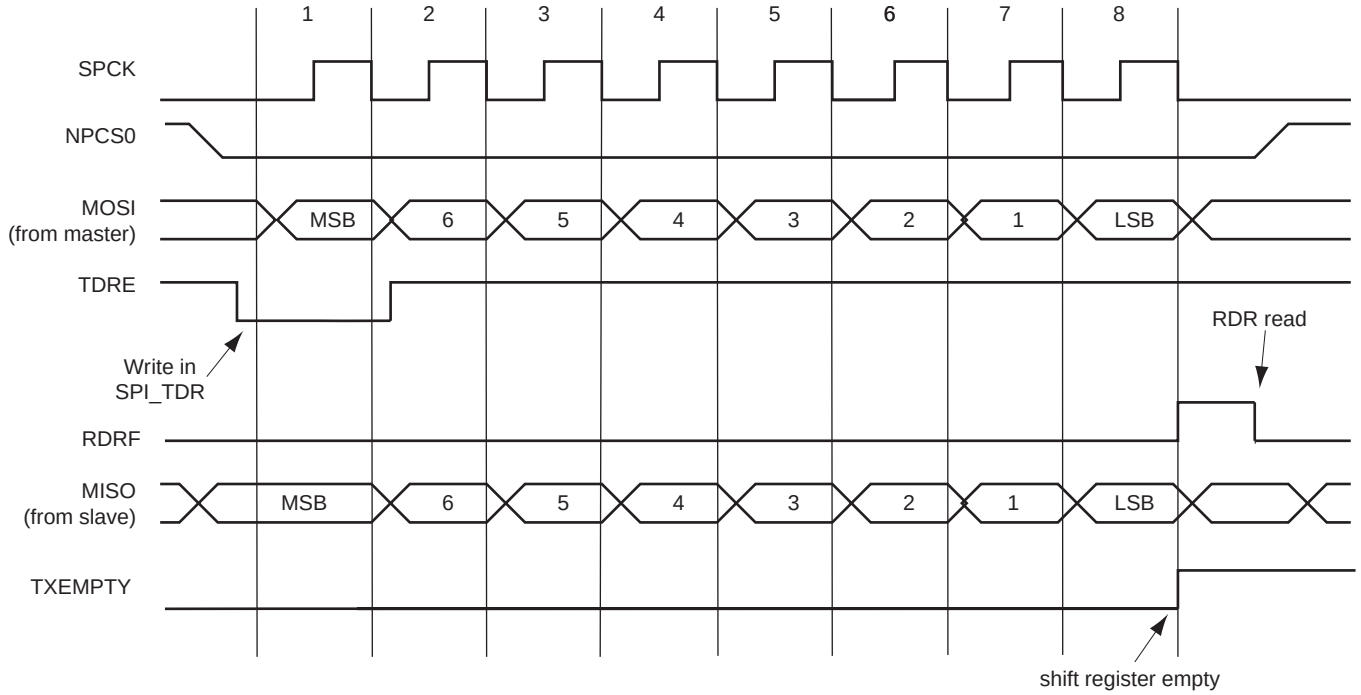
Figure 40-7. Master Mode Flow Diagram



Downloaded from Arrow.com.

Figure 40-8 shows the behavior of Transmit Data Register Empty (TDRE), Receive Data Register (RDRF) and Transmission Register Empty (TXEMPTY) status flags within the SPI_SR during an 8-bit data transfer in Fixed mode without the DMA involved.

Figure 40-8. Status Register Flags Behavior



40.7.3.3 Clock Generation

The SPI Baud rate clock is generated by dividing the peripheral clock by a value between 1 and 255.

If the SCBR field in the SPI_CSR is programmed to 1, the operating baud rate is peripheral clock (see Section 56. "Electrical Characteristics" for the SPCK maximum frequency). Triggering a transfer while SCBR is at 0 can lead to unpredictable results.

At reset, SCBR is 0 and the user has to program it to a valid value before performing the first transfer.

The divisor can be defined independently for each chip select, as it has to be programmed in the SCBR field. This allows the SPI to automatically adapt the baud rate for each interfaced peripheral without reprogramming.

40.7.3.4 Transfer Delays

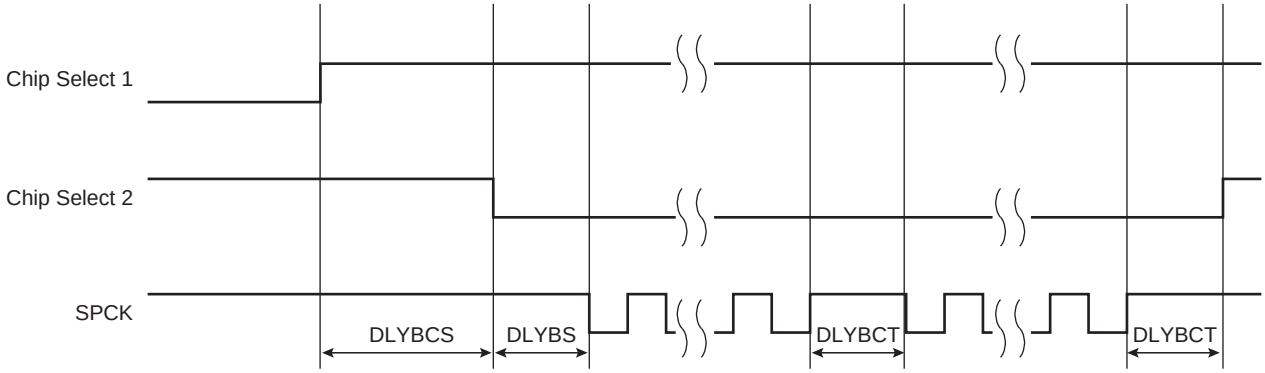
Figure 40-9 shows a chip select transfer change and consecutive transfers on the same chip select. Three delays can be programmed to modify the transfer waveforms:

- Delay between the chip selects—programmable only once for all chip selects by writing the DLYBCS field in the SPI_MR. The SPI slave device deactivation delay is managed through DLYBCS. If there is only one SPI slave device connected to the master, the DLYBCS field does not need to be configured. If several slave devices are connected to a master, DLYBCS must be configured depending on the highest deactivation delay. Refer to Section 56.13.1.5 and Section 56.14.1.5 .
- Delay before SPCK—independently programmable for each chip select by writing the DLYBS field. The SPI slave device activation delay is managed through DLYBS. Refer to Section 56.13.1.5 and Section 56.14.1.5 to define DLYBS.
- Delay between consecutive transfers—independently programmable for each chip select by writing the DLYBCT field. The time required by the SPI slave device to process received data is managed through DLYBCT. This time depends on the SPI slave system activity.

Downloaded from Arrow.com.

These delays allow the SPI to be adapted to the interfaced peripherals and their speed and bus release time.

Figure 40-9. Programmable Delays



40.7.3.5 Peripheral Selection

The serial peripherals are selected through the assertion of the NPCS0 to NPCS3 signals. By default, all NPCS signals are high before and after each transfer.

- **Fixed Peripheral Select Mode**: SPI exchanges data with only one peripheral.
  Fixed Peripheral Select mode is enabled by clearing the PS bit in the SPI_MR. In this case, the current peripheral is defined by the PCS field in the SPI_MR and the PCS field in the SPI_TDR has no effect.
- **Variable Peripheral Select Mode:** Data can be exchanged with more than one peripheral without having to reprogram the NPCS field in the SPI_MR.
  Variable Peripheral Select mode is enabled by setting the PS bit in the SPI_MR. The PCS field in the SPI_TDR is used to select the current peripheral. This means that the peripheral selection can be defined for each new data. The value to write in the SPI_TDR has the following format:

  [xxxxxxx(7-bit) + LASTXFER(1-bit)(1)+ xxxx(4-bit) + PCS (4-bit) + DATA (8 to 16-bit)] with PCS equals the chip select to assert, as defined in Section 40.8.4 “SPI Transmit Data Register” and LASTXFER bit at 0 or 1 depending on the CSAAT bit.

Note: 1. Optional

CSAAT, LASTXFER and CSNAAT bits are discussed in Section 40.7.3.9 “Peripheral Deselection with DMA”.

If LASTXFER is used, the command must be issued after writing the last character. Instead of LASTXFER, the user can use the SPIDIS command. After the end of the DMA transfer, it is necessary to wait for the TXEMPTY flag and then write SPIDIS into the SPI Control Register (SPI_CR). This does not change the configuration register values). The NPCS is disabled after the last character transfer. Then, another DMA transfer can be started if the SPIEN has previously been written in the SPI_CR.

40.7.3.6 SPI Direct Access Memory Controller (DMAC)

In both Fixed and Variable modes, the Direct Memory Access Controller (DMAC) can be used to reduce processor overhead.

The fixed peripheral selection allows buffer transfers with a single peripheral. Using the DMAC is an optimal means, as the size of the data transfer between the memory and the SPI is either 8 bits or 16 bits. However, if the peripheral selection is modified, the SPI_MR must be reprogrammed.

The variable peripheral selection allows buffer transfers with multiple peripherals without reprogramming the SPI_MR. Data written in the SPI_TDR is 32 bits wide and defines the real data to be transmitted and the destination peripheral. Using the DMAC in this mode requires 32-bit wide buffers, with the data in the LSBs and the

Downloaded from Arrow.com.

PCS and LASTXFER fields in the MSBs. However, the SPI still controls the number of bits (8 to 16) to be transferred through MISO and MOSI lines with the chip select configuration registers. This is not the optimal means in terms of memory size for the buffers, but it provides a very effective means to exchange data with several peripherals without any intervention of the processor.

40.7.3.7 Peripheral Chip Select Decoding

The user can program the SPI to operate with up to 15 slave peripherals by decoding the four chip select lines, NPCS0 to NPCS3 with an external decoder/demultiplexer (refer to Figure 40-10). This can be enabled by setting the PCSDEC bit in the SPI_MR.

When operating without decoding, the SPI makes sure that in any case only one chip select line is activated, i.e., one NPCS line driven low at a time. If two bits are defined low in a PCS field, only the lowest numbered chip select is driven low.

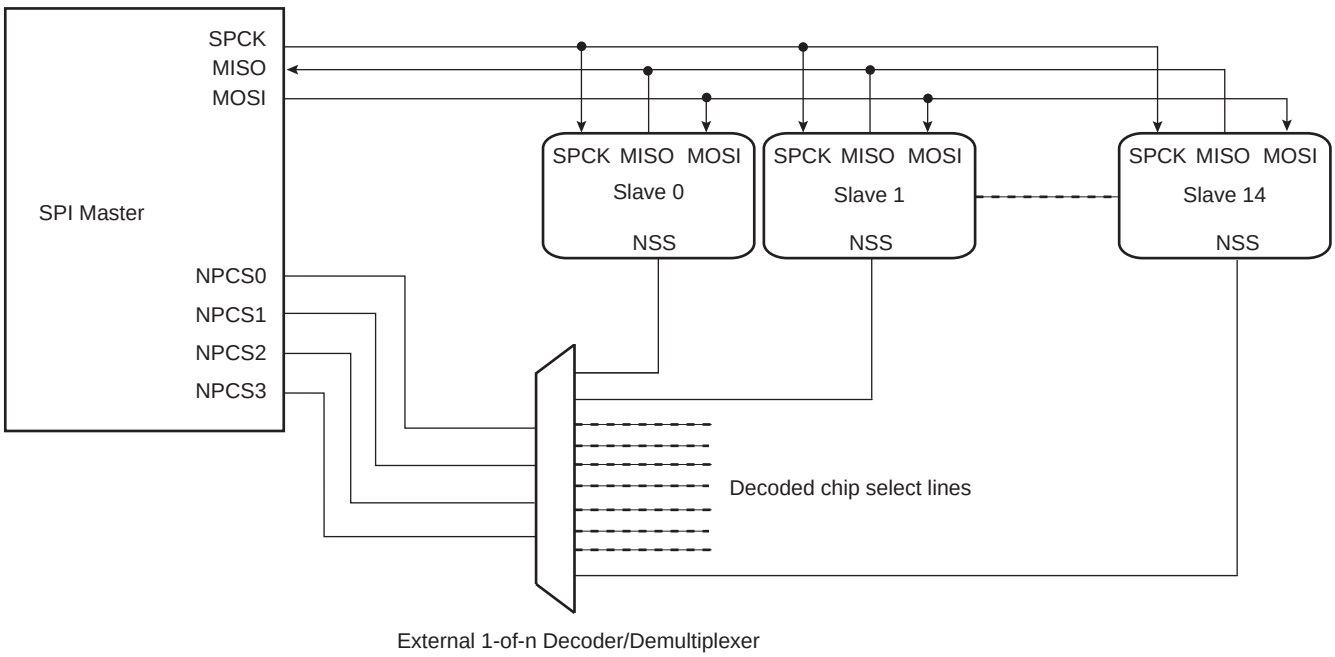
When operating with decoding, the SPI directly outputs the value defined by the PCS field on the NPCS lines of either SPI_MR or SPI_TDR (depending on PS).

As the SPI sets a default value of 0xF on the chip select lines (i.e., all chip select lines at 1) when not processing any transfer, only 15 peripherals can be decoded.

The SPI has four chip select registers (SPI_CSR0...SPI_CSR3). As a result, when external decoding is activated, each NPCS chip select defines the characteristics of up to four peripherals. As an example, SPI_CRS0 defines the characteristics of the externally decoded peripherals 0 to 3, corresponding to the PCS values 0x0 to 0x3. Consequently, the user has to make sure to connect compatible peripherals on the decoded chip select lines 0 to 3, 4 to 7, 8 to 11 and 12 to 14. Figure 40-10 shows this type of implementation.

If the CSAAT bit is used, with or without the DMAC, the Mode Fault detection for NPCS0 line must be disabled. This is not needed for all other chip select lines since Mode Fault detection is only on NPCS0.

Figure 40-10. Chip Select Decoding Application Block Diagram: Single Master/Multiple Slave Implementation



40.7.3.8 Peripheral Deselection without DMA

During a transfer of more than one unit of data on a chip select without the DMA, the SPI_TDR is loaded by the processor, the TDRE flag rises as soon as the content of the SPI_TDR is transferred into the internal Shift register. When this flag is detected high, the SPI_TDR can be reloaded. If this reload by the processor occurs before the

Downloaded from Arrow.com.

end of the current transfer and if the next transfer is performed on the same chip select as the current transfer, the chip select is not de-asserted between the two transfers. But depending on the application software handling the SPI status register flags (by interrupt or polling method) or servicing other interrupts or other tasks, the processor may not reload the SPI_TDR in time to keep the chip select active (low). A null DLYBCT value (delay between consecutive transfers) in the SPI_CSR, gives even less time for the processor to reload the SPI_TDR. With some SPI slave peripherals, if the chip select line must remain active (low) during a full set of transfers, communication errors can occur.

To facilitate interfacing with such devices, the chip select registers [SPI_CSR0...SPI_CSR3] can be programmed with the Chip Select Active After Transfer (CSAAT) bit at 1. This allows the chip select lines to remain in their current state (low = active) until a transfer to another chip select is required. Even if the SPI_TDR is not reloaded, the chip select remains active. To de-assert the chip select line at the end of the transfer, the Last Transfer (LASTXFER) bit in SPI_CR must be set after writing the last data to transmit into SPI_TDR.

40.7.3.9 Peripheral Deselection with DMA

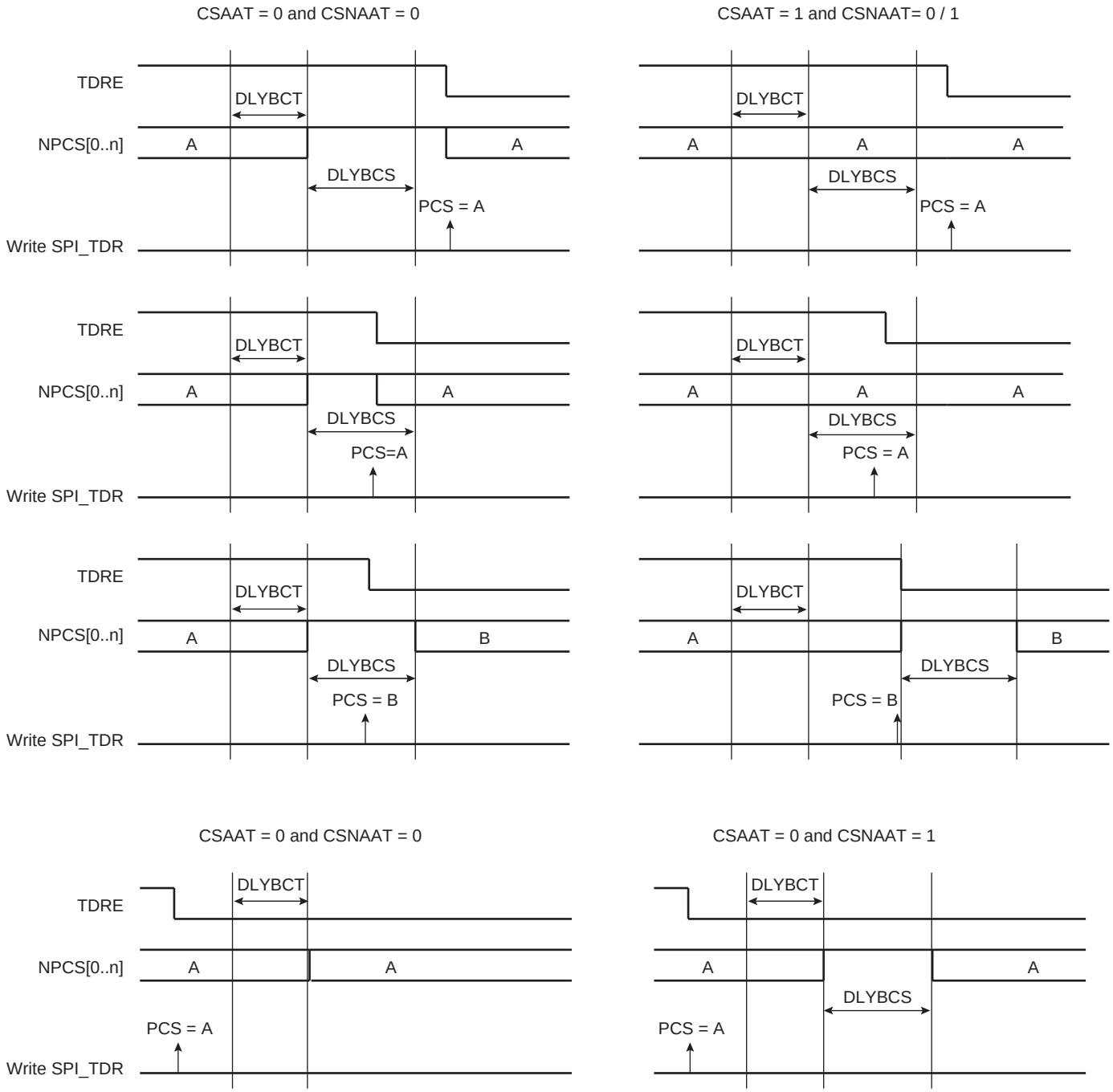
DMA provides faster reloads of the SPI_TDR compared to software. However, depending on the system activity, it is not guaranteed that the SPI_TDR is written with the next data before the end of the current transfer. Consequently, data can be lost by the de-assertion of the NPCS line for SPI slave peripherals requiring the chip select line to remain active between two transfers. The only way to guarantee a safe transfer in this case is the use of the CSAAT and LASTXFER bits.

When the CSAAT bit is configured to 0, the NPCS does not rise in all cases between two transfers on the same peripheral. During a transfer on a chip select, the TDRE flag rises as soon as the content of the SPI_TDR is transferred into the internal shift register. When this flag is detected, the SPI_TDR can be reloaded. If this reload occurs before the end of the current transfer and if the next transfer is performed on the same chip select as the current transfer, the chip select is not de-asserted between the two transfers. This can lead to difficulties to interface with some serial peripherals requiring the chip select to be de-asserted after each transfer. To facilitate interfacing with such devices, the SPI_CSR can be programmed with the Chip Select Not Active After Transfer (CSNAAT) bit at 1. This allows the chip select lines to be de-asserted systematically during a time “DLYBCS” (the value of the CSNAAT bit is processed only if the CSAAT bit is configured to 0 for the same chip select).

Figure 40-11 shows different peripheral deselection cases and the effect of the CSAAT and CSNAAT bits.

Downloaded from Arrow.com.

Figure 40-11. Peripheral Deselection



40.7.3.10 Mode Fault Detection

The SPI has the capability to operate in multi-master environment. Consequently, the NPCS0/NSS line must be monitored. If one of the masters on the SPI bus is currently transmitting, the NPCS0/NSS line is low and the SPI must not transmit any data. A mode fault is detected when the SPI is programmed in Master mode and a low level is driven by an external master on the NPCS0/NSS signal. In multi-master environment, NPCS0, MOSI, MISO and SPCK pins must be configured in open drain (through the PIO controller). When a mode fault is detected, the SPI_SR.MODF bit is set until SPI_SR is read and the SPI is automatically disabled until it is re-enabled by setting the SPI_CR.SPIEN bit.

By default, the mode fault detection is enabled. The user can disable it by setting the SPI_MR.MODFDIS bit.

Downloaded from Arrow.com.

40.7.4 SPI Slave Mode

When operating in Slave mode, the SPI processes data bits on the clock provided on the SPI clock pin (SPCK).

The SPI waits until NSS goes active before receiving the serial clock from an external master. When NSS falls, the clock is validated and the data is loaded in the SPI_RDR depending on the BITS field configured in SPI_CSR0. These bits are processed following a phase and a polarity defined respectively by the NCPHA and CPOL bits in SPI_CSR0. Note that the fileds BITS, CPOL and NCPHA of the other chip select registers (SPI_CSR1...SPI_CSR3) have no effect when the SPI is programmed in Slave mode.

The bits are shifted out on the MISO line and sampled on the MOSI line.

Note: For more information on the BITS field, see also the note below the SPI_CSRx register bitmap (Section 40.8.9 “SPI Chip Select Register”).

When all bits are processed, the received data is transferred in the SPI_RDR and the RDRF bit rises. If the SPI_RDR has not been read before new data is received, the Overrun Error Status (OVRES) bit in the SPI_SR is set. As long as this flag is set, data is loaded in the SPI_RDR. The user must read SPI_SR to clear the OVRES bit.

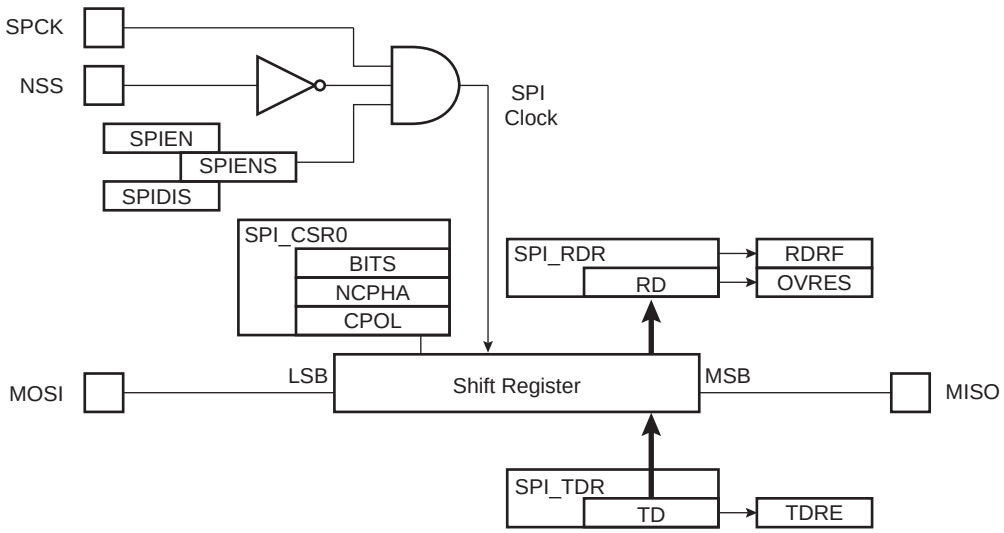
When a transfer starts, the data shifted out is the data present in the Shift register. If no data has been written in the SPI_TDR, the last data received is transferred. If no data has been received since the last reset, all bits are transmitted low, as the Shift register resets to 0.

When a first data is written in the SPI_TDR, it is transferred immediately in the Shift register and the TDRE flag rises. If new data is written, it remains in the SPI_TDR until a transfer occurs, i.e., NSS falls and there is a valid clock on the SPCK pin. When the transfer occurs, the last data written in the SPI_TDR is transferred in the Shift register and the TDRE flag rises. This enables frequent updates of critical variables with single transfers.

Then, new data is loaded in the Shift register from the SPI_TDR. If no character is ready to be transmitted, i.e., no character has been written in the SPI_TDR since the last load from the SPI_TDR to the Shift register, the SPI_TDR is retransmitted. In this case the Underrun Error Status Flag (UNDES) is set in the SPI_SR.

Figure 40-12 shows a block diagram of the SPI when operating in Slave mode.

Figure 40-12. Slave Mode Functional Block Diagram



Downloaded from Arrow.com.

40.7.5 Register Write Protection

To prevent any single software error from corrupting SPI behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the SPI Write Protection Mode Register (SPI_WPMR).

If a write access to a write-protected register is detected, the WPVS flag in the SPI Write Protection Status Register (SPI_WPSR) is set and the WPVSRC field indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading SPI_WPSR.

The following registers can be write-protected:

- SPI Mode Register
- SPI Chip Select Register

Downloaded from Arrow.com.

40.8 Serial Peripheral Interface (SPI) User Interface

In the “Offset” column of Table 40-5, ‘CS_number’ denotes the chip select number.

Table 40-5. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	SPI_CR	Write-only	–
0x04	Mode Register	SPI_MR	Read/Write	0x0
0x08	Receive Data Register	SPI_RDR	Read-only	0x0
0x0C	Transmit Data Register	SPI_TDR	Write-only	–
0x10	Status Register	SPI_SR	Read-only	0x0
0x14	Interrupt Enable Register	SPI_IER	Write-only	–
0x18	Interrupt Disable Register	SPI_IDR	Write-only	–
0x1C	Interrupt Mask Register	SPI_IMR	Read-only	0x0
0x20–0x2C	Reserved	–	–	–
0x30 + (CS_number * 0x04)	Chip Select Register	SPI_CSR	Read/Write	0x0
0x40–0x48	Reserved	–	–	–
0x4C–0xE0	Reserved	–	–	–
0xE4	Write Protection Mode Register	SPI_WPMR	Read/Write	0x0
0xE8	Write Protection Status Register	SPI_WPSR	Read-only	0x0
0xEC–0xF8	Reserved	–	–	–
0xFC	Reserved	–	–	–

Downloaded from Arrow.com.

40.8.1 SPI Control Register

Name: SPI_CR

Address: 0x40008000 (0), 0x40058000 (1)

Access: Write-only

31	30	29	28	27	26	25	24
FIFODIS	FIFOEN	–	–	–	–	–	LASTXFER
23	22	21	20	19	18	17	16
–	–	–	–	–	–	RXFCLR	TXFCLR
15	14	13	12	11	10	9	8
–	–	–	REQCLR	–	–	–	–
7	6	5	4	3	2	1	0
SWRST	–	–	–	–	–	SPIDIS	SPIEN

- **SPIEN: SPI Enable**

0: No effect.

1: Enables the SPI to transfer and receive data.

- **SPIDIS: SPI Disable**

0: No effect.

1: Disables the SPI.

All pins are set in Input mode after completion of the transmission in progress, if any.

If a transfer is in progress when SPIDIS is set, the SPI completes the transmission of the shifter register and does not start any new transfer, even if the SPI_THR is loaded.

Note: If both SPIEN and SPIDIS are equal to one when the SPI_CR is written, the SPI is disabled.

- **SWRST: SPI Software Reset**

0: No effect.

1: Reset the SPI. A software-triggered hardware reset of the SPI interface is performed.

The SPI is in Slave mode after software reset.

- **REQCLR: Request to Clear the Comparison Trigger**

0: No effect.

1: Restarts the comparison trigger to enable SPI_RDR loading.

- **TXFCLR: Transmit FIFO Clear**

0: No effect.

1: Clears the Transmit FIFO, Transmit FIFO will become empty.

- **RXFCLR: Receive FIFO Clear**

0: No effect.

1: Clears the Receive FIFO, Receive FIFO will become empty.

Downloaded from Arrow.com.

- **LASTXFER: Last Transfer**

0: No effect.

1: The current NPCS is de-asserted after the character written in TD has been transferred. When SPI_CSRx.CSAAT is set, the communication with the current serial peripheral can be closed by raising the corresponding NPCS line as soon as TD transfer is completed.

Refer to Section 40.7.3.5 “Peripheral Selection” for more details.

- **FIFOEN: FIFO Enable**

0: No effect.

1: Enables the Transmit and Receive FIFOs

- **FIFODIS: FIFO Disable**

0: No effect.

1: Disables the Transmit and Receive FIFOs

Downloaded from Arrow.com.

40.8.2 SPI Mode Register

Name: SPI_MR

Address: 0x40008004 (0), 0x40058004 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
DLYBCS							
23	22	21	20	19	18	17	16
–	–	–	–	PCS			
15	14	13	12	11	10	9	8
–	–	–	-	–	–	–	–
7	6	5	4	3	2	1	0
LLB	–	WDRBT	MODFDIS	–	PCSDEC	PS	MSTR

This register can only be written if the WPEN bit is cleared in the SPI Write Protection Mode Register.

- **MSTR: Master/Slave Mode**

0: SPI is in Slave mode

1: SPI is in Master mode

- **PS: Peripheral Select**

0: Fixed Peripheral Select

1: Variable Peripheral Select

- **PCSDEC: Chip Select Decode**

0: The chip select lines are directly connected to a peripheral device.

1: The four NPCS chip select lines are connected to a 4-bit to 16-bit decoder.

When PCSDEC = 1, up to 15 chip select signals can be generated with the four NPCS lines using an external 4-bit to 16-bit decoder. The chip select registers define the characteristics of the 15 chip selects, with the following rules:

SPI_CSR0 defines peripheral chip select signals 0 to 3.

SPI_CSR1 defines peripheral chip select signals 4 to 7.

SPI_CSR2 defines peripheral chip select signals 8 to 11.

SPI_CSR3 defines peripheral chip select signals 12 to 14.

- **MODFDIS: Mode Fault Detection**

0: Mode fault detection enabled

1: Mode fault detection disabled

- **WDRBT: Wait Data Read Before Transfer**

0: No Effect. In Master mode, a transfer can be initiated regardless of the SPI_RDR state.

1: In Master mode, a transfer can start only if the SPI_RDR is empty, i.e., does not contain any unread data. This mode prevents overrun error in reception.

Downloaded from Arrow.com.

- **LLB: Local Loopback Enable**

0: Local loopback path disabled.

1: Local loopback path enabled.

LLB controls the local loopback on the data shift register for testing in Master mode only (MISO is internally connected on MOSI).

- **PCS: Peripheral Chip Select**

This field is only used if fixed peripheral select is active (PS = 0).

If SPI_MR.PCSDEC = 0:

PCS = xxx0 NPCS[3:0] = 1110

PCS = xx01 NPCS[3:0] = 1101

PCS = x011 NPCS[3:0] = 1011

PCS = 0111 NPCS[3:0] = 0111

PCS = 1111 forbidden (no peripheral is selected)

(x = don't care)

If SPI_MR.PCSDEC = 1:

NPCS[3:0] output signals = PCS.

- **DLYBCS: Delay Between Chip Selects**

This field defines the delay between the inactivation and the activation of NPCS. The DLYBCS time guarantees non-overlapping chip selects and solves bus contentions in case of peripherals having long data float times.

If DLYBCS is lower than 6, six peripheral clock periods are inserted by default.

Otherwise, the following equation determines the delay:

:

$$\text{Delay Between Chip Selects} = \frac{\text{DLYBCS}}{f_{\text{peripheral clock}}}$$

Downloaded from Arrow.com.

40.8.3 SPI Receive Data Register

Name: SPI_RDR

Address: 0x40008008 (0), 0x40058008 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	PCS			

15	14	13	12	11	10	9	8
RD							

7	6	5	4	3	2	1	0
RD							

- **RD: Receive Data**

Data received by the SPI Interface is stored in this register in a right-justified format. Unused bits are read as zero.

- **PCS: Peripheral Chip Select**

In Master mode only, these bits indicate the value on the NPCS pins at the end of a transfer. Otherwise, these bits are read as zero.

Note: When using Variable Peripheral Select mode (PS = 1 in SPI_MR), it is mandatory to set the SPI_MR.WDRBT bit if the PCS field must be processed in SPI_RDR.

Downloaded from Arrow.com.

40.8.4 SPI Transmit Data Register

Name: SPI_TDR

Address: 0x4000800C (0), 0x4005800C (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	LASTXFER

23	22	21	20	19	18	17	16
–	–	–	–	PCS			

15	14	13	12	11	10	9	8
TD							

7	6	5	4	3	2	1	0
TD							

- **TD: Transmit Data**

Data to be transmitted by the SPI Interface is stored in this register. Information to be transmitted must be written to the transmit data register in a right-justified format.

- **PCS: Peripheral Chip Select**

This field is only used if variable peripheral select is active (PS = 1).

If SPI_MR.PCSDEC = 0:

PCS = xxx0 NPCS[3:0] = 1110

PCS = xx01 NPCS[3:0] = 1101

PCS = x011 NPCS[3:0] = 1011

PCS = 0111 NPCS[3:0] = 0111

PCS = 1111 forbidden (no peripheral is selected)

(x = don’t care)

If SPI_MR.PCSDEC = 1:

NPCS[3:0] output signals = PCS.

- **LASTXFER: Last Transfer**

0: No effect

1: The current NPCS is de-asserted after the transfer of the character written in TD. When SPI_CSRx.CSAAT is set, the communication with the current serial peripheral can be closed by raising the corresponding NPCS line as soon as TD transfer is completed.

This field is only used if variable peripheral select is active (SPI_MR.PS = 1).

Downloaded from Arrow.com.

40.8.5 SPI Status Register

Name: SPI_SR

Address: 0x40008010 (0), 0x40058010 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	SPIENS
15	14	13	12	11	10	9	8
–	–	–	–	–	UNDES	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
–	–	–	–	OVRES	MODF	TDRE	RDRF

- **RDRF: Receive Data Register Full (cleared by reading SPI_RDR)**

0: No data has been received since the last read of SPI_RDR.

1: Data has been received and the received data has been transferred from the shift register to SPI_RDR since the last read of SPI_RDR.

- **TDRE: Transmit Data Register Empty (cleared by writing SPI_TDR)**

0: Data has been written to SPI_TDR and not yet transferred to the shift register.

1: The last data written in the SPI_TDR has been transferred to the shift register.

TDRE equals zero when the SPI is disabled or at reset. The SPI enable command sets this bit to 1.

- **MODF: Mode Fault Error (cleared on read)**

0: No mode fault has been detected since the last read of SPI_SR.

1: A mode fault occurred since the last read of SPI_SR.

- **OVRES: Overrun Error Status (cleared on read)**

0: No overrun has been detected since the last read of SPI_SR.

1: An overrun has occurred since the last read of SPI_SR.

An overrun occurs when SPI_RDR is loaded at least twice from the shift register since the last read of the SPI_RDR.

- **NSSR: NSS Rising (cleared on read)**

0: No rising edge detected on NSS pin since the last read of SPI_SR.

1: A rising edge occurred on NSS pin since the last read of SPI_SR.

- **TXEMPTY: Transmission Registers Empty (cleared by writing SPI_TDR)**

0: As soon as data is written in SPI_TDR.

1: SPI_TDR and internal shift register are empty. If a transfer delay has been defined, TXEMPTY is set after the end of this delay.

- **UNDES: Underrun Error Status (Slave mode only) (cleared on read)**

0: No underrun has been detected since the last read of SPI_SR.

1: A transfer starts whereas no data has been loaded in SPI_TDR.

Downloaded from Arrow.com.

- **SPIENS: SPI Enable Status**

0: SPI is disabled.

1: SPI is enabled.

Downloaded from Arrow.com.

40.8.6 SPI Interrupt Enable Register

Name: SPI_IER

Address: 0x40008014 (0), 0x40058014 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	UNDES	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
–	–	–	–	OVRES	MODF	TDRE	RDRF

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

- **RDRF: Receive Data Register Full Interrupt Enable**
- **TDRE: SPI Transmit Data Register Empty Interrupt Enable**
- **MODF: Mode Fault Error Interrupt Enable**
- **OVRES: Overrun Error Interrupt Enable**
- **NSSR: NSS Rising Interrupt Enable**
- **TXEMPTY: Transmission Registers Empty Enable**
- **UNDES: Underrun Error Interrupt Enable**

Downloaded from Arrow.com.

40.8.7 SPI Interrupt Disable Register

Name: SPI_IDR

Address: 0x40008018 (0), 0x40058018 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	UNDES	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
–	–	–	–	OVRES	MODF	TDRE	RDRF

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

- **RDRF: Receive Data Register Full Interrupt Disable**
- **TDRE: SPI Transmit Data Register Empty Interrupt Disable**
- **MODF: Mode Fault Error Interrupt Disable**
- **OVRES: Overrun Error Interrupt Disable**
- **NSSR: NSS Rising Interrupt Disable**
- **TXEMPTY: Transmission Registers Empty Disable**
- **UNDES: Underrun Error Interrupt Disable**

Downloaded from Arrow.com.

40.8.8 SPI Interrupt Mask Register

Name: SPI_IMR

Address: 0x4000801C (0), 0x4005801C (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	UNDES	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
–	–	–	–	OVRES	MODF	TDRE	RDRF

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

- **RDRF: Receive Data Register Full Interrupt Mask**
- **TDRE: SPI Transmit Data Register Empty Interrupt Mask**
- **MODF: Mode Fault Error Interrupt Mask**
- **OVRES: Overrun Error Interrupt Mask**
- **NSSR: NSS Rising Interrupt Mask**
- **TXEMPTY: Transmission Registers Empty Mask**
- **UNDES: Underrun Error Interrupt Mask**

Downloaded from Arrow.com.

40.8.9 SPI Chip Select Register

Name: SPI_CSRx [x=0..3]

Address: 0x40008030 (0), 0x40058030 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
DLYBCT							
23	22	21	20	19	18	17	16
DLYBS							
15	14	13	12	11	10	9	8
SCBR							
7	6	5	4	3	2	1	0
BITS				CSAAT	CSNAAT	NCPHA	CPOL

This register can only be written if the WPEN bit is cleared in the SPI Write Protection Mode Register.

Note: SPI_CSRx registers must be written even if the user wants to use the default reset values. The BITS field is not updated with the translated value unless the register is written.

- **CPOL: Clock Polarity**

0: The inactive state value of SPCK is logic level zero.

1: The inactive state value of SPCK is logic level one.

CPOL is used to determine the inactive state value of the serial clock (SPCK). It is used with NCPHA to produce the required clock/data relationship between master and slave devices.

- **NCPHA: Clock Phase**

0: Data is changed on the leading edge of SPCK and captured on the following edge of SPCK.

1: Data is captured on the leading edge of SPCK and changed on the following edge of SPCK.

NCPHA determines which edge of SPCK causes data to change and which edge causes data to be captured. NCPHA is used with CPOL to produce the required clock/data relationship between master and slave devices.

- **CSNAAT: Chip Select Not Active After Transfer (Ignored if CSAAT = 1)**

0: The Peripheral Chip Select Line does not rise between two transfers if the SPI_TDR is reloaded before the end of the first transfer and if the two transfers occur on the same chip select.

1: The Peripheral Chip Select Line rises systematically after each transfer performed on the same slave. It remains inactive after the end of transfer for a minimal duration of:

$$\frac{DLYBCS}{f_{peripheral\ clock}}$$ (If field DLYBCS is lower than 6, a minimum of six periods is introduced.)

- **CSAAT: Chip Select Active After Transfer**

0: The Peripheral Chip Select Line rises as soon as the last transfer is achieved.

1: The Peripheral Chip Select Line does not rise after the last transfer is achieved. It remains active until a new transfer is requested on a different chip select.

Downloaded from Arrow.com.

- **BITS: Bits Per Transfer**

(See the note below the register bitmap.)

The BITS field determines the number of data bits transferred. Reserved values should not be used.

Value	Name	Description
0	8_BIT	8 bits for transfer
1	9_BIT	9 bits for transfer
2	10_BIT	10 bits for transfer
3	11_BIT	11 bits for transfer
4	12_BIT	12 bits for transfer
5	13_BIT	13 bits for transfer
6	14_BIT	14 bits for transfer
7	15_BIT	15 bits for transfer
8	16_BIT	16 bits for transfer
9	–	Reserved
10	–	Reserved
11	–	Reserved
12	–	Reserved
13	–	Reserved
14	–	Reserved
15	–	Reserved

- **SCBR: Serial Clock Bit Rate**

In Master mode, the SPI Interface uses a modulus counter to derive the SPCK bit rate from the peripheral clock. The bit rate is selected by writing a value from1 to 255 in the SCBR field. The following equation determines the SPCK bit rate:

SCBR = $f_{peripheral\ clock}$ / SPCK Bit Rate

Programming the SCBR field to 0 is forbidden. Triggering a transfer while SCBR is at 0 can lead to unpredictable results.

If BRSRCCLK = 1 in SPI_MR, SCBR must be programmed with a value greater than 1.

At reset, SCBR is 0 and the user has to program it at a valid value before performing the first transfer.

Note: If one of the SCBR fields in SPI_CSRx is set to 1, the other SCBR fields in SPI_CSRx must be set to 1 as well, if they are used to process transfers. If they are not used to transfer data, they can be set at any value.

- **DLYBS: Delay Before SPCK**

This field defines the delay from NPCS falling edge (activation) to the first valid SPCK transition.

When DLYBS = 0, the delay is half the SPCK clock period.

Otherwise, the following equation determines the delay:

DLYBS = Delay Before SPCK × $f_{peripheral\ clock}$

- **DLYBCT: Delay Between Consecutive Transfers**

This field defines the delay between two consecutive transfers with the same peripheral without removing the chip select. The delay is always inserted after each transfer and before removing the chip select if needed.

When DLYBCT = 0, no delay between consecutive transfers is inserted and the clock keeps its duty cycle over the character transfers.

Downloaded from Arrow.com.

Otherwise, the following equation determines the delay:

DLYBCT = Delay Between Consecutive Transfers × $f_{peripheral\ clock}$ / 32

Downloaded from Arrow.com.

40.8.10 SPI Write Protection Mode Register

Name: SPI_WPMR

Address: 0x400080E4 (0), 0x400580E4 (1)

Access: Read/Write.

31	30	29	28	27	26	25	24
WPKEY							
23	22	21	20	19	18	17	16
WPKEY							
15	14	13	12	11	10	9	8
WPKEY							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x535049 (“SPI” in ASCII)

1: Enables the write protection if WPKEY corresponds to 0x535049 (“SPI” in ASCII)

See Section 40.7.5 “Register Write Protection” for the list of registers that can be write-protected.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x535049	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

Downloaded from Arrow.com.

40.8.11 SPI Write Protection Status Register

Name: SPI_WPSR

Address: 0x400080E8 (0), 0x400580E8 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
WPVSRC							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

- **WPVS: Write Protection Violation Status**

0: No write protection violation has occurred since the last read of SPI_WPSR.

1: A write protection violation has occurred since the last read of SPI_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protection Violation Source**

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

Downloaded from Arrow.com.

41. Quad SPI Interface (QSPI)

41.1 Description

The Quad SPI Interface (QSPI) is a synchronous serial data link that provides communication with external devices in Master mode.

The QSPI can be used in SPI mode to interface to serial peripherals (such as ADCs, DACs, LCD controllers, CAN controllers and sensors), or in Serial Memory mode to interface to serial Flash memories.

The QSPI allows the system to execute code directly from a serial Flash memory (XIP) without code shadowing to RAM. The serial Flash memory mapping is seen in the system as other memories (ROM, SRAM, DRAM, embedded Flash memory, etc.).

With the support of the Quad SPI protocol, the QSPI allows the system to use high-performance serial Flash memories which are small and inexpensive, in place of larger and more expensive parallel Flash memories.

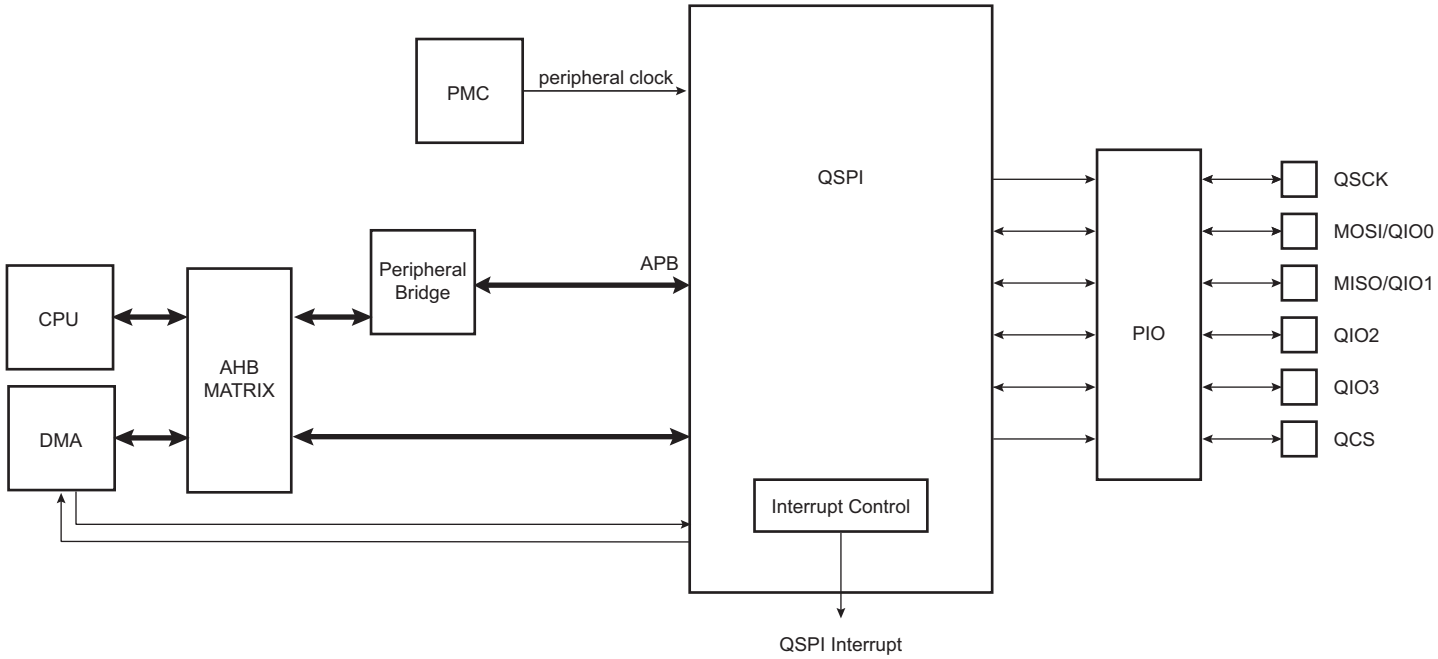
41.2 Embedded Characteristics

- Master SPI Interface
  - Programmable Clock Phase and Clock Polarity
  - Programmable Transfer Delays Between Consecutive Transfers, Between Clock and Data, Between Deactivation and Activation of Chip Select
- SPI Mode
  - Interface to Serial Peripherals such as ADCs, DACs, LCD Controllers, CAN Controllers and Sensors
  - 8-bit/16-bit/32-bit Programmable Data Length
- Serial Memory Mode
  - Interface to Serial Flash Memories Operating in Single-bit SPI, Dual SPI and Quad SPI
  - Supports “Execute In Place” (XIP)— Code Execution by the System Directly from a Serial Flash Memory
  - Flexible Instruction Register for Compatibility with All Serial Flash Memories
  - 32-bit Address Mode (default is 24-bit address) to Support Serial Flash Memories Larger than 128 Mbit
  - Continuous Read Mode
  - Scrambling/Unscrambling “On-The-Fly”
- Connection to DMA Channel Capabilities Optimizes Data Transfers
  - One channel for the Receiver, One Channel for the Transmitter
- Register Write Protection

Downloaded from Arrow.com.

41.3 Block Diagram

Figure 41-1. Block Diagram



41.4 Signal Description

Table 41-1. Signal Description

Pin Name	Pin Description	Type
QSCK	Serial Clock	Output
MOSI (QIO0) (1) (2)	Data Output (Data Input Output 0)	Output (Input/Output)
MISO (QIO1) (1) (2)	Data Input (Data Input Output 1)	Input (Input/Output)
QIO2 (3)	Data Input Output 2	Input/Output
QIO3 (3)	Data Input Output 3	Input/Output
QCS	Peripheral Chip Select	Output

Notes:
1. MOSI and MISO are used for single-bit SPI operation.
2. QIO0–QIO1 are used for Dual SPI operation.
3. QIO0–QIO3 are used for Quad SPI operation.

Downloaded from Arrow.com.

41.5 Product Dependencies

41.5.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the QSPI pins to their peripheral functions.

Table 41-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
QSPI	QCS	PA11	A
QSPI	QIO0	PA13	A
QSPI	QIO1	PA12	A
QSPI	QIO2	PA17	A
QSPI	QIO3	PD31	A
QSPI	QSCK	PA14	A

41.5.2 Power Management

The QSPI may be clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the QSPI clock.

41.5.3 Interrupt Sources

The QSPI has an interrupt line connected to the Interrupt Controller. Handling the QSPI interrupt requires programming the interrupt controller before configuring the QSPI.

Table 41-3. Peripheral IDs

Instance	ID
QSPI	43

41.5.4 Direct Memory Access Controller (DMA)

The QSPI can be used in conjunction with the Direct Memory Access Controller (DMA) in order to reduce processor overhead. For a full description of the DMA, refer to Section 35. ”DMA Controller (XDMAC)”.

Note: DMA write accesses must be 32-bit aligned. If a single byte is to be written in a 32-bit word, the rest of the word must be filled with ones.

Downloaded from Arrow.com.

41.6 Functional Description

41.6.1 Serial Clock Baud Rate

The QSPI baud rate clock is generated by dividing the peripheral clock by a value between 1 and 256.

41.6.2 Serial Clock Phase and Polarity

Four combinations of polarity and phase are available for data transfers. The clock polarity is programmed with the CPOL bit in the QSPI Serial Clock register (QSPI_SCR). The CPHA bit in the QSPI_SCR programs the clock phase. These two parameters determine the edges of the clock signal on which data is driven and sampled. Each of the two parameters has two possible states, resulting in four possible combinations that are incompatible with one another. Thus, the interfaced slave must use the same parameter values to communicate.

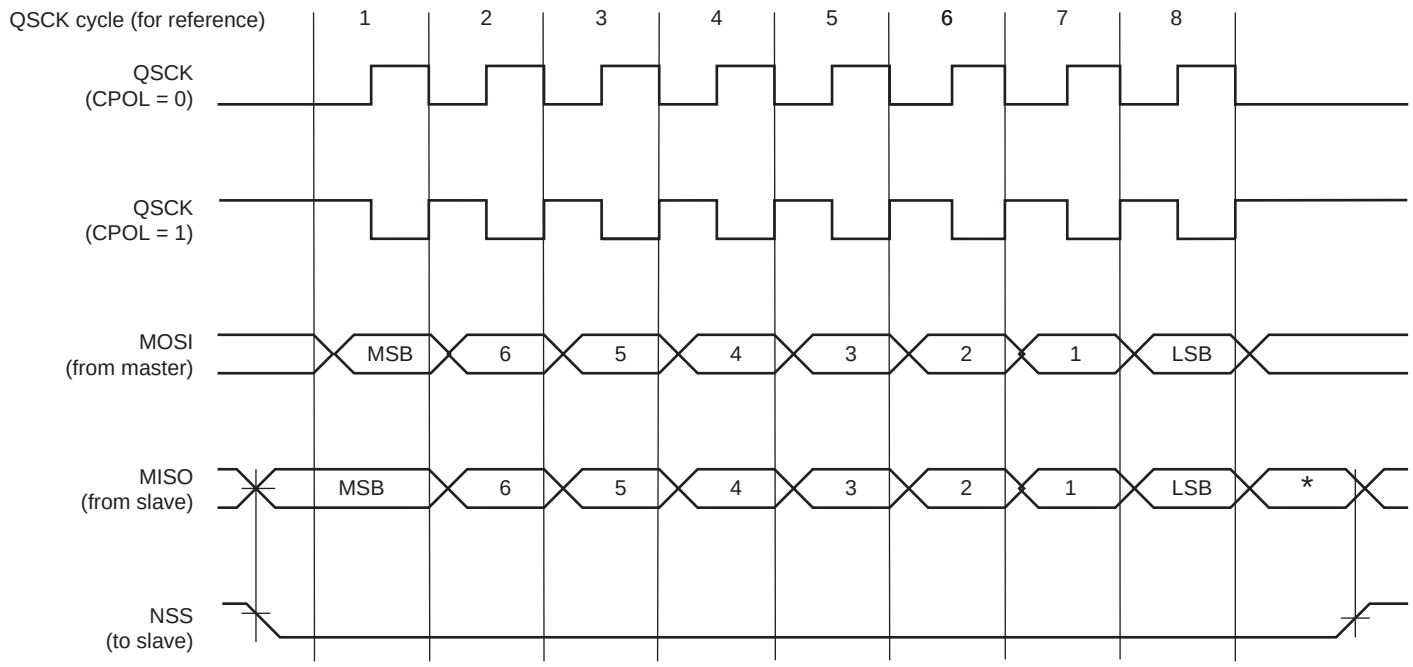
Table 41-4 shows the four modes and corresponding parameter settings.

Table 41-4. QSPI Bus Clock Modes

QSPI Clock Mode	QSPI_SCR.CPOL	QSPI_SCR.CPHA	Shift QSCK Edge	Capture QSCK Edge	QSCK Inactive Level
0	0	0	Falling	Rising	Low
1	0	1	Rising	Falling	Low
2	1	0	Rising	Falling	High
3	1	1	Falling	Rising	High

Figure 41-2 and Figure 41-3 show examples of data transfers.

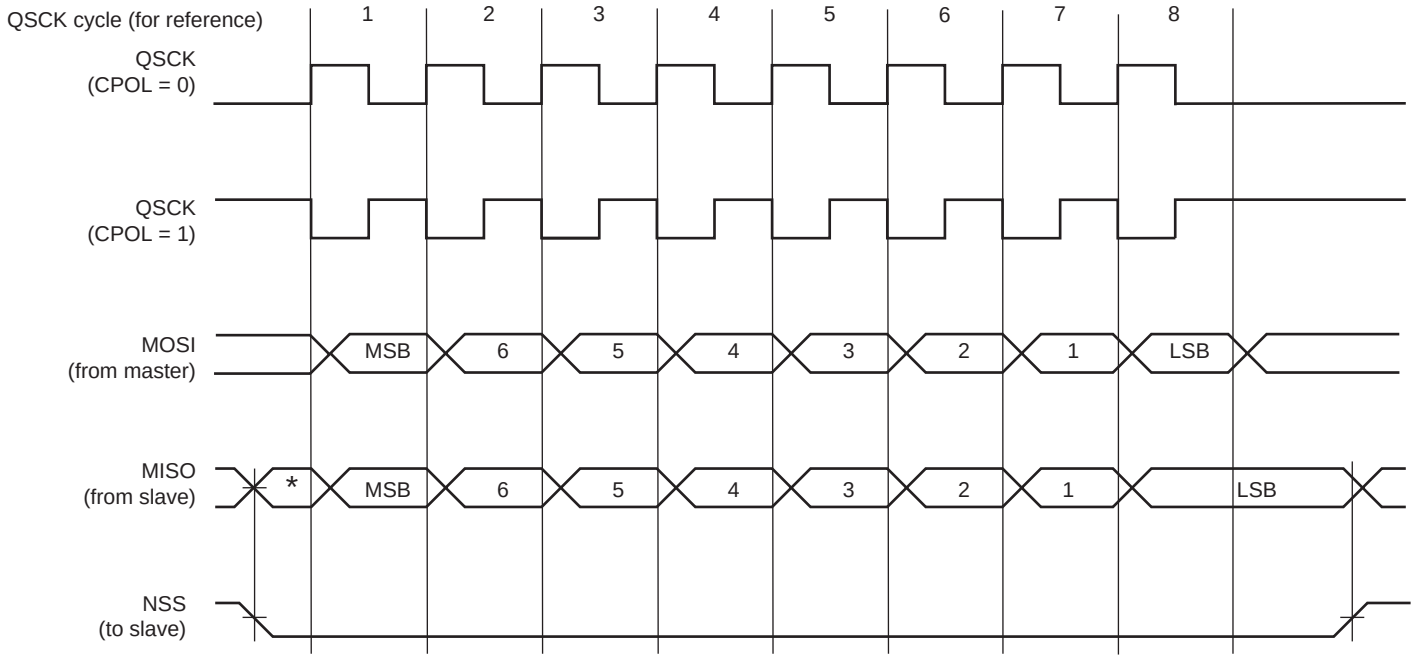
Figure 41-2. QSPI Transfer Format (QSPI_SCR.CPHA = 0, 8 bits per transfer)



* Not defined, but normally MSB of previous character received.

Downloaded from Arrow.com.

Figure 41-3. QSPI Transfer Format (QSPI_SCR.CPHA = 1, 8 bits per transfer)



* Not defined but normally LSB of previous character transmitted.

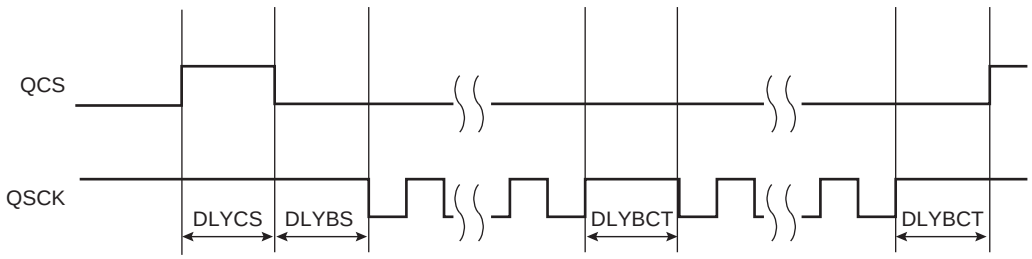
41.6.3 Transfer Delays

Figure 41-4 shows several consecutive transfers while the chip select is active. Three delays can be programmed to modify the transfer waveforms:

- The delay between the deactivation and the activation of QCS, programmed by writing the field DLYCS. Allows to adjust the minimum time of QCS at high level.
- The delay before QSCK, programmed by writing the field DLYBS. Allows the start of QSCK to be delayed after the chip select has been asserted.
- The delay between consecutive transfers, programmed by writing the DLYBCT field. Allows insertion of a delay between two consecutive transfers. In Serial Memory mode, this delay is not programmable and DLYBCT is ignored. In that mode, DLYBCT must be set to 0.

These delays allow the QSPI to be adapted to the interfaced peripherals and their speed and bus release time.

Figure 41-4. Programmable Delays



41.6.4 QSPI SPI Mode

In SPI mode, the QSPI acts as a regular SPI Master.

To activate this mode, bit SMM must be cleared in the Mode register (QSPI_MR).

Downloaded from Arrow.com.

41.6.4.1 SPI Mode Operations

The QSPI in standard SPI mode operates on the clock generated by the internal programmable baud rate generator. It fully controls the data transfers to and from the slave connected to the SPI bus. The QSPI drives the chip select line to the slave (QCS) and the serial clock signal (QSCK).

The QSPI features two holding registers, the Transmit Data register (QSPI_TDR) and the Receive Data register (QSPI_RDR), and a single internal shift register. The holding registers maintain the data flow at a constant rate.

After enabling the QSPI, a data transfer begins when the processor writes to the QSPI_TDR. The written data is immediately transferred to the internal shift register and transfer on the SPI bus starts. While the data in the internal shift register is shifted on the MOSI line, the MISO line is sampled and shifted to the internal shift register. Receiving data cannot occur without transmitting data. If receiving mode is not needed, for example when communicating with a slave receiver only (such as an LCD), the receive status flags in the Status register (QSPI_SR) can be discarded.

If new data is written in QSPI_TDR during the transfer, it is retained there until the current transfer is completed. Then, the received data is transferred from the internal shift register to the QSPI_RDR, the data in QSPI_TDR is loaded in the internal shift register and a new transfer starts.

The transfer of a data written in QSPI_TDR in the internal shift register is indicated by the Transmit Data Register Empty (TDRE) bit in the QSPI_SR. When new data is written in QSPI_TDR, this bit is cleared. The TDRE bit is used to trigger the Transmit DMA channel.

The end of transfer is indicated by the TXEMPTY flag in the QSPI_SR. If a transfer delay (DLYBCT) is greater than 0 for the last transfer, TXEMPTY is set after the completion of this delay. The peripheral clock can be switched off at this time.

The transfer of received data from the internal shift register in QSPI_RDR is indicated by the Receive Data Register Full (RDRF) bit in the QSPI_SR. When the received data is read, the RDRF bit is cleared.

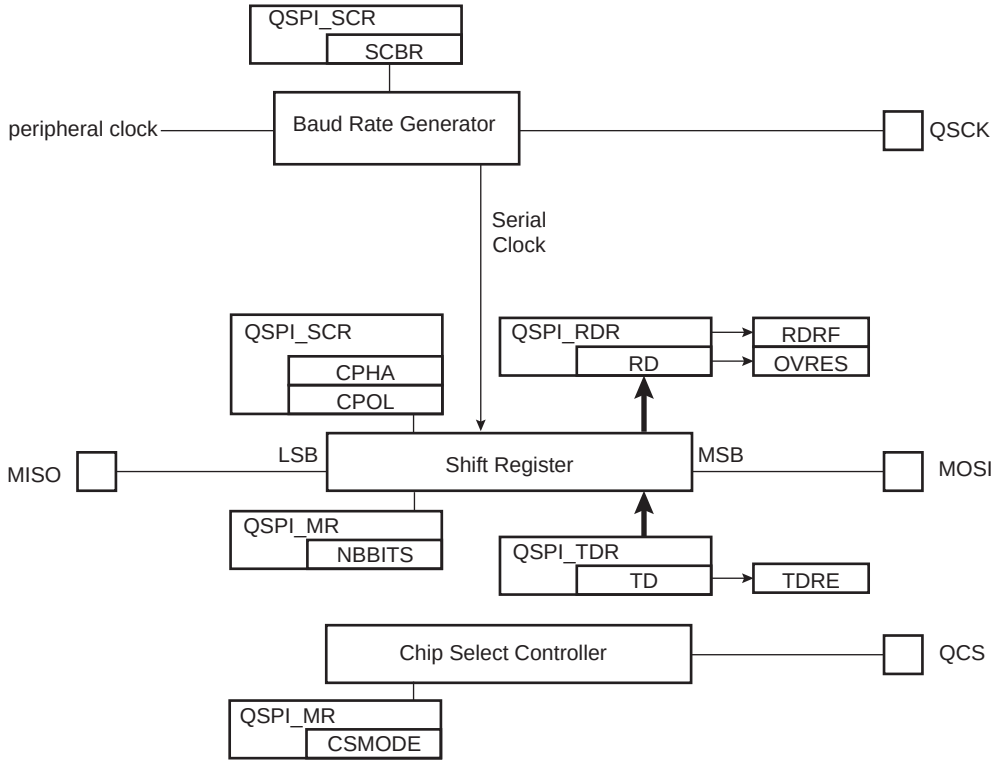
If the QSPI_RDR has not been read before new data is received, the Overrun Error Status (OVRES) bit in QSPI_SR is set. As long as this flag is set, data is loaded in QSPI_RDR. The user must read the QSPI_SR to clear the OVRES bit.

Figure 41-5 shows a block diagram of the SPI when operating in Master mode. Figure 41-6 shows a flow chart describing how transfers are handled.

Downloaded from Arrow.com.

41.6.4.2 SPI Mode Block Diagram

Figure 41-5. SPI Mode Block Diagram



Downloaded from Arrow.com.

41.6.4.3 SPI Mode Flow Diagram

Figure 41-6. SPI Mode Flow Diagram

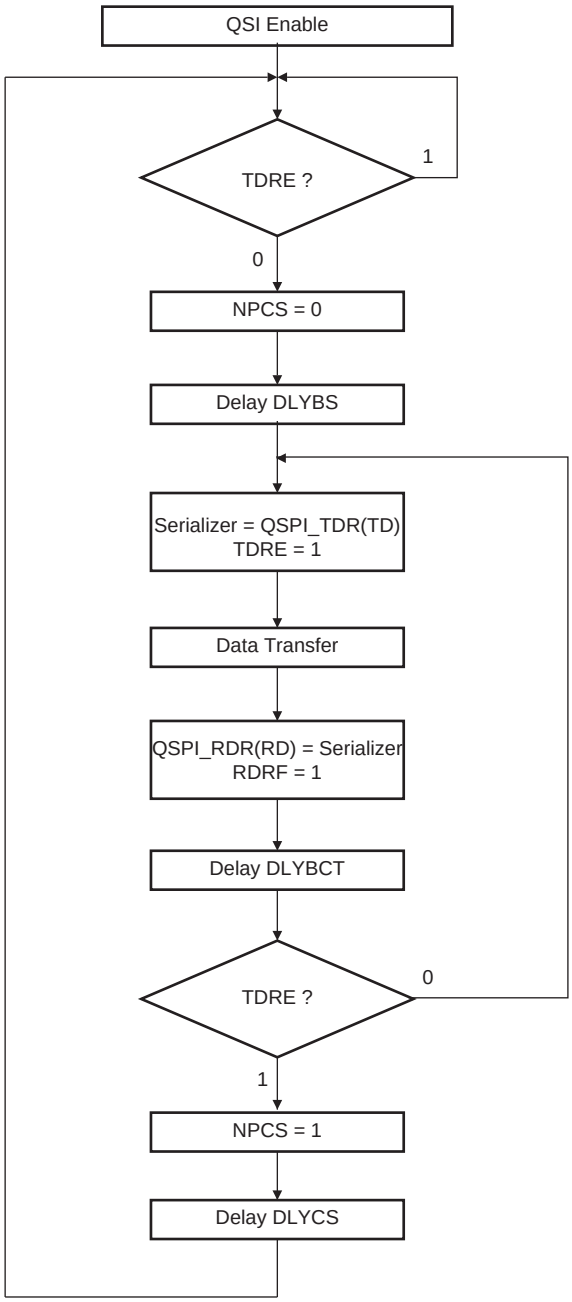
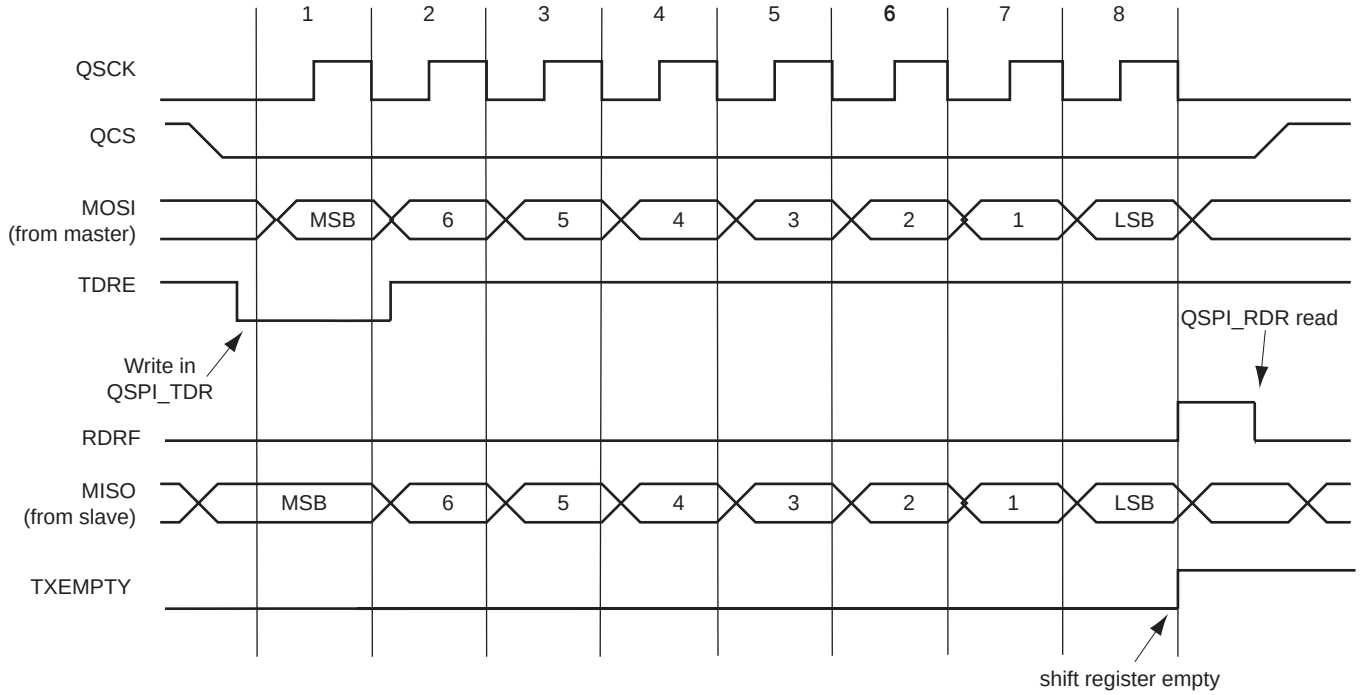


Figure 41-7 shows Transmit Data Register Empty (TDRE), Receive Data Register Full (RDRF) and Transmission Register Empty (TXEMPTY) status flags behavior within the QSPI_SR during an 8-bit data transfer in Fixed mode, without DMA.

Downloaded from Arrow.com.

Figure 41-7. Status Register Flags Behavior



41.6.4.4 Peripheral Deselection without DMA

During a transfer of more than one data on a Chip Select without the DMA, the QSPI_TDR is loaded by the processor and the flag TDRE rises as soon as the content of the QSPI_TDR is transferred into the internal shift register. When this flag is detected high, the QSPI_TDR can be reloaded. If this reload by the processor occurs before the end of the current transfer and if the next transfer is performed on the same chip select as the current transfer, the Chip Select is not de-asserted between the two transfers. Depending on the application software handling the QSPI_SR flags (by interrupt or polling method) or servicing other interrupts or other tasks, the processor may not reload the QSPI_TDR in time to keep the chip select active (low). A null Delay Between Consecutive Transfer (DLYBCT) value in the QSPI_MR gives even less time for the processor to reload the QSPI_TDR. With some SPI slave peripherals, requiring the chip select line to remain active (low) during a full set of transfers may lead to communication errors.

To facilitate interfacing with such devices, the QSPI_MR can be programmed with the CSMODE field at 1. This allows the chip select lines to remain in their current state (low = active) until the end of transfer is indicated by the Last Transfer (LASTXFER) bit in the Control register (QSPI_CR). Even if the QSPI_TDR is not reloaded, the chip select remains active. To have the chip select line rise at the end of the last data transfer, the LASTXFER bit in the QSPI_CR must be set at the same time or after writing the last data to transmit into the QSPI_TDR.

41.6.4.5 Peripheral Deselection with DMA

When the DMA Controller is used, the Chip Select line remains low during the transfer since the TDRE flag is managed by the DMA itself. Reloading the QSPI_TDR by the DMA is done as soon as the TDRE flag is set to one. In this case, setting the CSMODE field to 1 might not be needed. However, when other DMA channels connected to other peripherals are also in use, the QSPI DMA could be delayed by another DMA with a higher priority on the bus. Having DMA buffers in slower memories like Flash memory or SDRAM compared to fast internal SRAM, may lengthen the reload time of the QSPI_TDR by the DMA as well. This means that the QSPI_TDR might not be reloaded in time to keep the chip select line low. In this case, the chip select line may toggle between data transfer and according to some SPI Slave devices, the communication might get lost. It may be necessary to configure the CSMODE field to 1.

Downloaded from Arrow.com.

When the CSMODE field is configured to 0, the QCS does not rise in all cases between two transfers on the same peripheral. During a transfer on a Chip Select, the flag TDRE rises as soon as the content of the QSPI_TDR is transferred into the internal shifter. When this flag is detected, the QSPI_TDR can be reloaded. If this reload occurs before the end of the current transfer and if the next transfer is performed on the same chip select as the current transfer, the Chip Select is not de-asserted between the two transfers. This might lead to difficulties for interfacing with some serial peripherals requiring the chip select to be de-asserted after each transfer. To facilitate interfacing with such devices, the QSPI_MR can be programmed with the CSMODE field at 2.

41.6.5 QSPI Serial Memory Mode

In Serial Memory mode, the QSPI acts as a serial Flash memory controller. The QSPI can be used to read data from the serial Flash memory allowing the CPU to execute code from it (XIP execute in place). The QSPI can also be used to control the serial Flash memory (Program, Erase, Lock, etc.) by sending specific commands. In this mode, the QSPI is compatible with single-bit SPI, Dual SPI and Quad SPI protocols.

To activate this mode, the bit SMM must be set in the QSPI_MR.

In Serial Memory mode, data cannot be transferred by the QSPI_TDR and the QSPI_RDR, but by writing or reading the QSPI memory space (0x8000_0000).

41.6.5.1 Instruction Frame

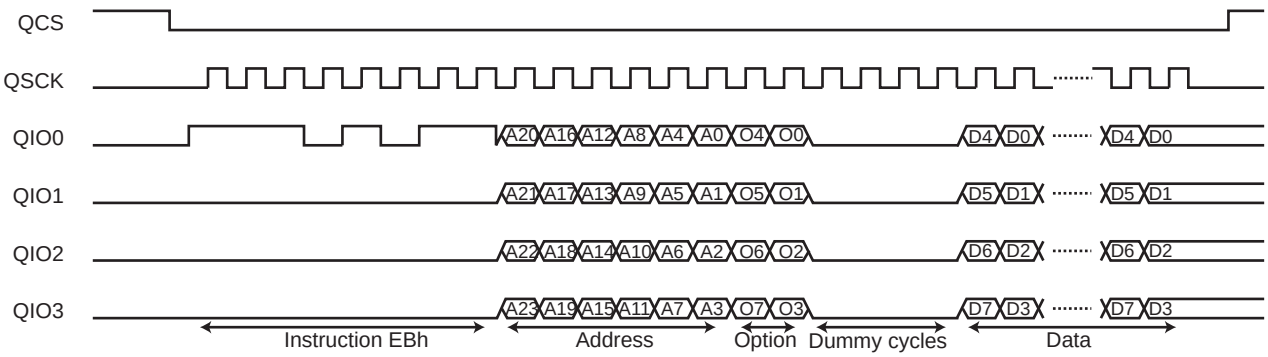
In order to control serial Flash memories, the QSPI is able to send instructions via the SPI bus (ex: READ, PROGRAM, ERASE, LOCK, etc.). Because the instruction set implemented in serial Flash memories is memory vendor dependant, the QSPI includes a complete Instruction Frame register (QSPI_IFR), which makes it very flexible and compatible with all serial Flash memories.

An instruction frame includes:

- An instruction code (size: 8 bits). The instruction is optional in some cases (see Section 41.6.5.4).
- An address (size: 24 bits or 32 bits). The address is optional but is required by instructions such as READ, PROGRAM, ERASE, LOCK. By default the address is 24 bits long, but it can be 32 bits long to support serial Flash memories larger than 128 Mbit (16 Mbyte).
- An option code (size: 1/2/4/8 bits). The option code is not required, but it is useful to activate the XIP mode or the Continuous Read mode (see Section 41.6.5.4) for READ instructions, in some serial Flash memory devices. These modes improve the data read latency.
- Dummy cycles. Dummy cycles are optional but required by some READ instructions.
- Data bytes are optional. Data bytes are present for data transfer instructions such as READ or PROGRAM.

The instruction code, the address/option and the data can be sent with Single-bit SPI, Dual SPI or Quad SPI protocols.

Figure 41-8. Instruction Frame



Downloaded from Arrow.com.

41.6.5.2 Instruction Frame Transmission

To send an instruction frame, the user must first configure the address to send by writing the field ADDR in the Instruction Address register (QSPI_IAR). This step is required if the instruction frame includes an address and no data. When data is present, the address of the instruction is defined by the address of the data accesses in the QSPI memory space, not by QSPI_IAR.

If the instruction frame includes the instruction code and/or the option code, the user must configure the instruction code and/or the option code to send by writing the fields INST and OPT in the Instruction Code register (QSPI_ICR).

Then, the user must write the Instruction Frame register (QSPI_IFR) to configure the instruction frame depending on which instruction must be sent. If the instruction frame does not include data, writing in this register triggers the send of the instruction frame in the QSPI. If the instruction frame includes data, the send of the instruction frame is triggered by the first data access in the QSPI memory space.

The instruction frame is configured by the following bits and fields of QSPI_IFR:

- WIDTH field—used to configure which data lanes are used to send the instruction code, the address, the option code and to transfer the data. It is possible to use two unidirectional data lanes (MISO-MOSI Single-bit SPI), two bidirectional data lanes (QIO0-QIO1 Dual SPI) or four bidirectional data lanes (QIO0–QIO3 Quad SPI).
- INSTEN bit—used to enable the send of an instruction code.
- ADDREN bit—used to enable the send of an address after the instruction code.
- OPTEN bit—used to enable the send of an option code after the address.
- DATAEN bit—used to enable the transfer of data (READ or PROGRAM instruction).
- OPTL field—used to configure the option code length. The value written in OPTL must be consistent with the value written in the field WIDTH. For example: OPTL = 0 (1-bit option code) is not consistent with WIDTH = 6 (option code sent with QuadSPI protocol, thus the minimum length of the option code is 4 bits).
- ADDRL bit—used to configure the address length.
- TFRTYP field—used to define which type of data transfer must be performed.
- NBDUM field—used to configure the number of dummy cycles when reading data from the serial Flash memory. Between the address/option and the data, with some instructions, dummy cycles are inserted by the serial Flash memory.

Refer to Section 41.7.12 ”QSPI Instruction Frame Register”.

If data transfer is enabled, the user can access the serial memory by reading or writing the QSPI memory space:

- To read in the serial memory, but not a memory data, for example a JEDEC-ID or the QSPI_SR, TFRTYP must be set to 0.
- To read in the serial memory, and particularly a memory data, TFRTYP must be set to 1.
- To write in the serial memory, but not a memory data, for example writing the configuration or the QSPI_SR, TFRTYP must be set to 2.
- If the user wants to write in the serial memory in particular to program a memory data, TFRTYP must be set to 3.

If TFRTYP has a value other than 1, the address sent in the instruction frame is the address of the first system bus accesses. The addresses of the next accesses are not used by the QSPI. At each system bus access, an SPI transfer is performed with the same size. For example, a halfword system bus access leads to a 16-bit SPI transfer, and a byte system bus access leads to an 8-bit SPI transfer.

If TFRTYP = 1, the address of the first instruction frame is the one of the first read access in the QSPI memory space. Each time the read accesses become non-sequential (addresses are not consecutive), a new instruction frame is sent with the last system bus access address. In this way, the system can read data at a random location in the serial memory. The size of the SPI transfers may differ from the size of the system bus read accesses.

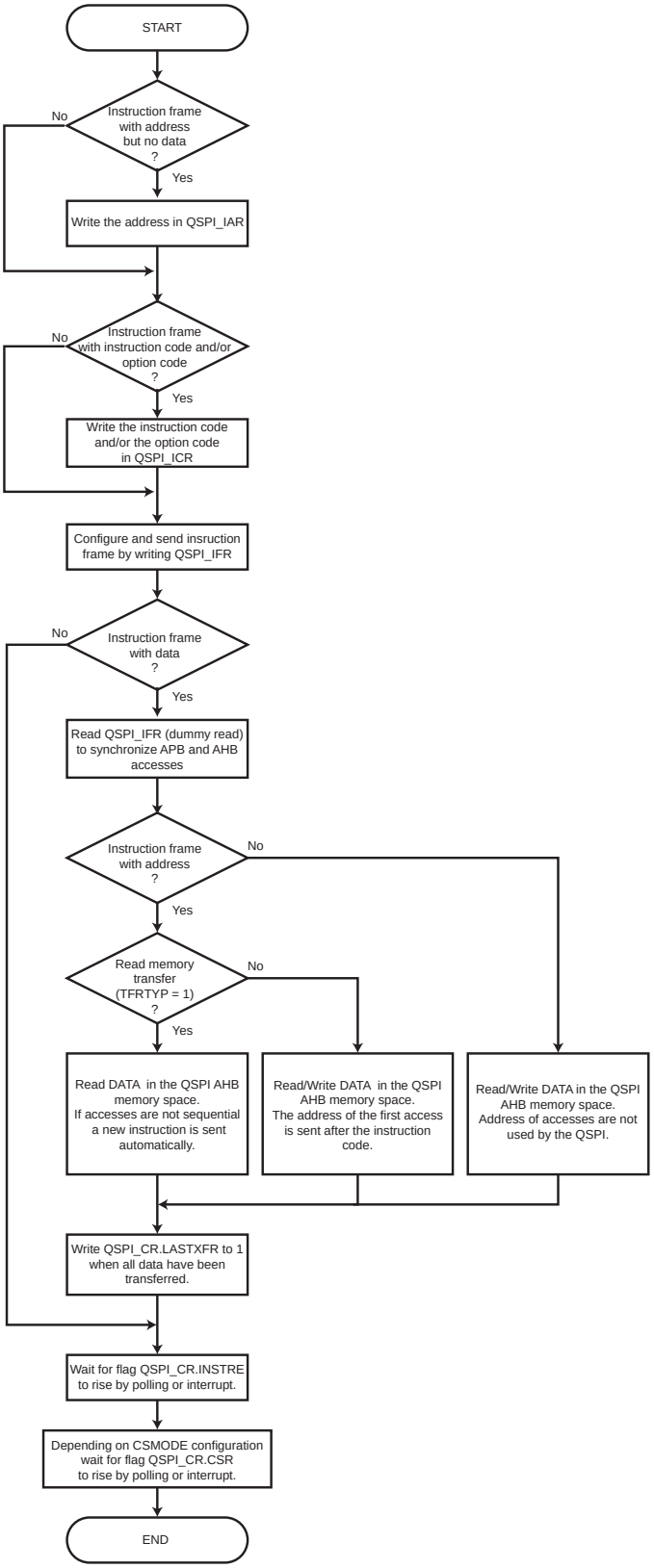
Downloaded from Arrow.com.

When data transfer is not enabled, the end of the instruction frame is indicated when QSPI_SR.INSTRE rises. (The QSPI_SR.CSR flag indicates when chip select rises. A delay between these flags may exist in case of high clock division or a high DLYBCT value). When data transfer is enabled, the user must indicate when the data transfer is completed in the QSPI memory space by setting QSPI_CR.LASTXFR. The end of the instruction frame is indicated when QSPI_SR.INSTRE rises.

Figure 41-9 illustrates instruction transmission management.

Downloaded from Arrow.com.

Figure 41-9. Instruction Transmission Flow Diagram



Downloaded from Arrow.com.

41.6.5.3 Read Memory Transfer

The user can access the data of the serial memory by sending an instruction with DATAEN = 1 and TFRTYP = 1 in QSPI_IFR.

In this mode the QSPI is able to read data at random address into the serial Flash memory, allowing the CPU to execute code directly from it (XIP execute-in-place).

In order to fetch data, the user must first configure the instruction frame by writing the QSPI_IFR. Then data can be read at any address in the QSPI address space mapping. The address of the system bus read accesses match the address of the data inside the serial Flash memory.

When Fetch mode is enabled, several instruction frames can be sent before writing the bit LASTXFR in the QSPI_CR. Each time the system bus read accesses become non- sequential (addresses are not consecutive), a new instruction frame is sent with the corresponding address.

41.6.5.4 Continuous Read Mode

The QSPI is compatible with the Continuous Read mode which is implemented in some serial Flash memories.

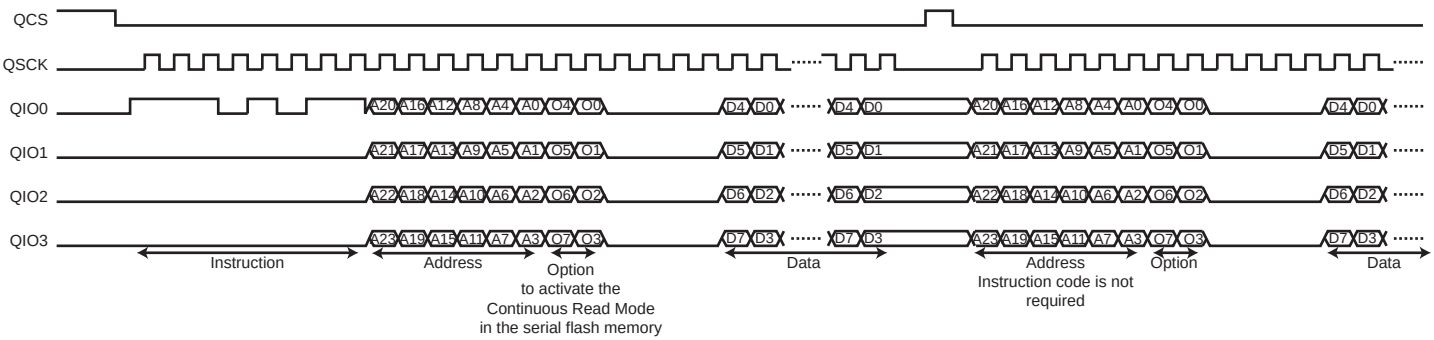
In Continuous Read mode, the instruction overhead is reduced by excluding the instruction code from the instruction frame. When the Continuous Read mode is activated in a serial Flash memory by a specific option code, the instruction code is stored in the memory. For the next instruction frames, the instruction code is not required as the memory uses the stored one.

In the QSPI, Continuous Read mode is used when reading data from the memory (TFRTYP = 1). The addresses of the system bus read accesses are often non-sequential and this leads to many instruction frames that have the same instruction code. By disabling the send of the instruction code, the Continuous Read mode reduces the access time of the data.

To be functional, this mode must be enabled in both the QSPI and the serial Flash memory. The Continuous Read mode is enabled in the QSPI by setting bit CRM in the QSPI_IFR (TFRTYP field value must equal 1). The Continuous Read mode is enabled in the serial Flash memory by sending a specific option code.

CAUTION: If the Continuous Read mode is not supported by the serial Flash memory or disabled, CRM bit must not be set, otherwise data read out the serial Flash memory is unpredictable.

Figure 41-10. Continuous Read Mode



Downloaded from Arrow.com.

41.6.5.5 Instruction Frame Transmission Examples

All waveforms in the following examples describe SPI transfers in SPI Clock mode 0 (QSPI_SCR.CPOL = 0 and QSPI_SCR.CPHA = 0; see Section 41.6.2 "Serial Clock Phase and Polarity").

All system bus accesses described below refer to the system bus address phase. System bus wait cycles and system bus data phases are not shown.

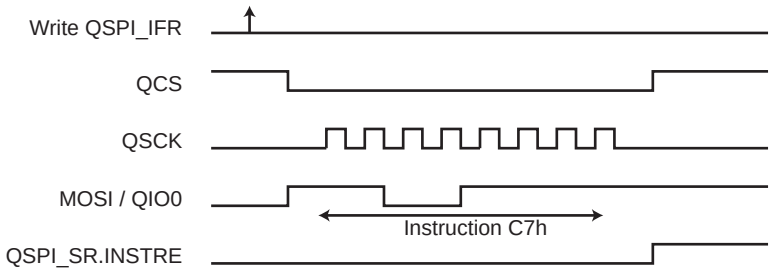
Example 1:

Instruction in Single-bit SPI, without address, without option, without data.

Command: CHIP ERASE (C7h).

- Write 0x0000_00C7 in QSPI_ICR.
- Write 0x0000_0010 in QSPI_IFR.
- Wait for INSTRE in QSPI_SR to rise.

Figure 41-11. Instruction Transmission Waveform 1



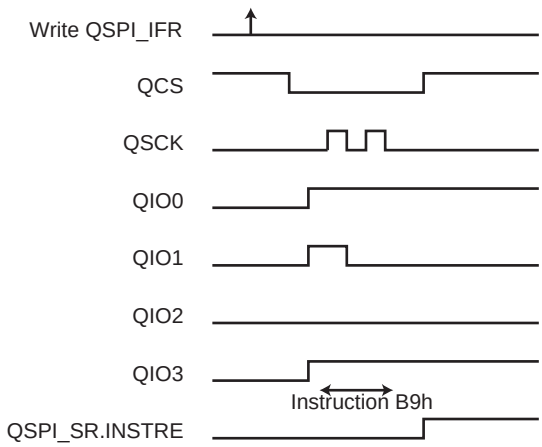
Example 2:

Instruction in Quad SPI, without address, without option, without data.

Command: POWER DOWN (B9h)

- Write 0x0000_00B9 in QSPI_ICR.
- Write 0x0000_0016 in QSPI_IFR.
- Wait for INSTRE in QSPI_SR to rise.

Figure 41-12. Instruction Transmission Waveform 2



Downloaded from Arrow.com.

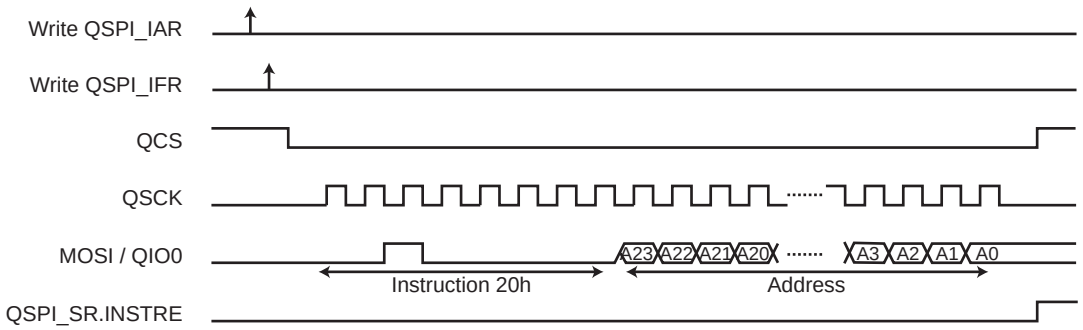
Example 3:

Instruction in Single-bit SPI, with address in Single-bit SPI, without option, without data.

Command: BLOCK ERASE (20h)

- Write the address (of the block to erase) in QSPI_AR.
- Write 0x0000_0020 in QSPI_ICR.
- Write 0x0000_0030 in QSPI_IFR.
- Wait for INSTRE in QSPI_SR to rise.

Figure 41-13. Instruction Transmission Waveform 3



Downloaded from Arrow.com.

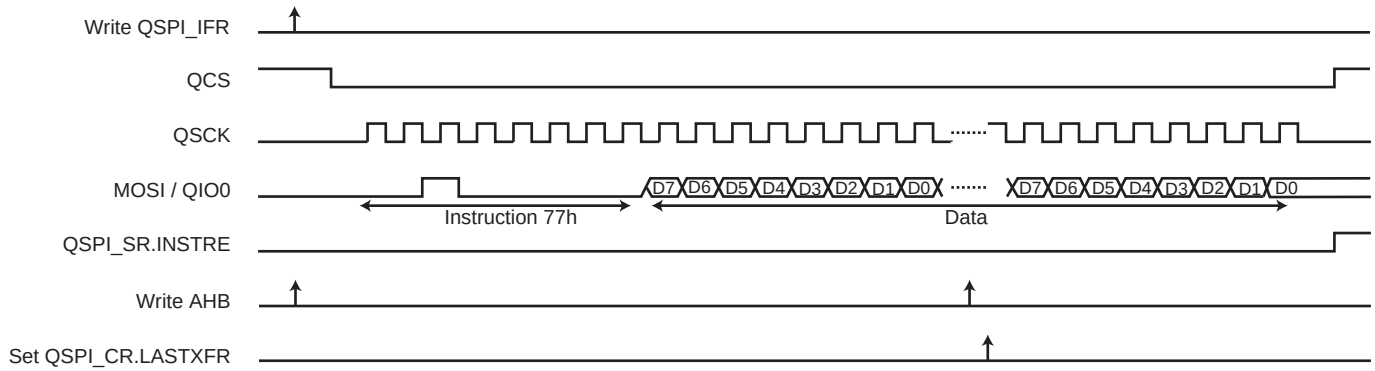
Example 4:

Instruction in Single-bit SPI, without address, without option, with data write in Single-bit SPI.

Command: SET BURST (77h)

- Write 0x0000_0077 in QSPI_ICR.
- Write 0x0000_2090 in QSPI_IFR.
- Read QSPI_IFR (dummy read) to synchronize system bus accesses.
- Write data in the system bus memory space (0x8000_0000).
  The address of system bus write accesses is not used.
- Set LASTXFR bit in QSPI_CR.
- Wait for INSTRE in QSPI_SR to rise.

Figure 41-14. Instruction Transmission Waveform 4



Downloaded from Arrow.com.

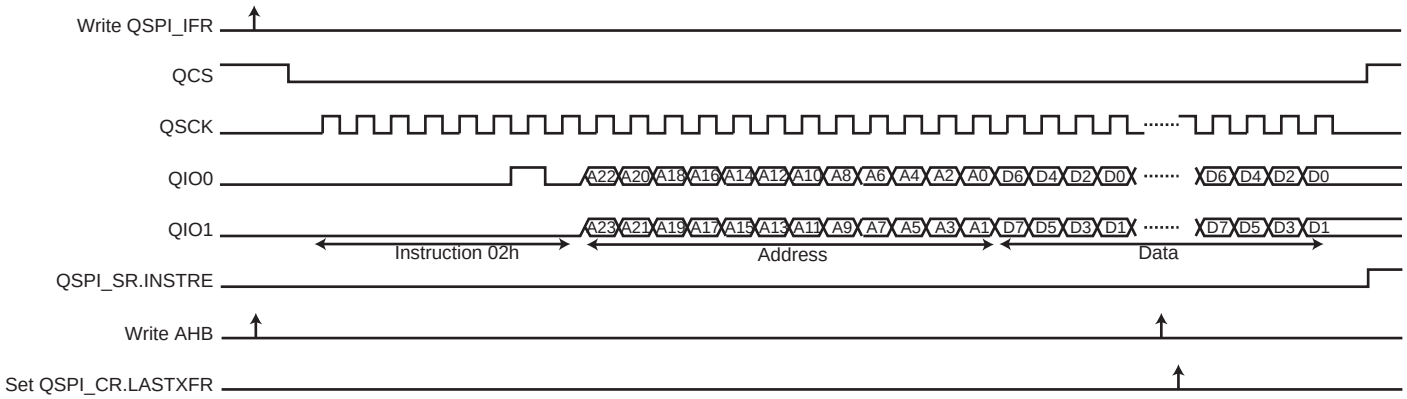
Example 5:

Instruction in Single-bit SPI, with address in Dual SPI, without option, with data write in Dual SPI.

Command: BYTE/PAGE PROGRAM (02h)

- Write 0x0000_0002 in QSPI_ICR.
- Write 0x0000_30B3 in QSPI_IFR.
- Read QSPI_IFR (dummy read) to synchronize system bus accesses.
- Write data in the QSPI system bus memory space (0x8000_0000).
  The address of the first system bus write access is sent in the instruction frame.
  The address of the next system bus write accesses is not used.
- Set LASTXFR bit in QSPI_CR.
- Wait for INSTRE in QSPI_SR to rise.

Figure 41-15. Instruction Transmission Waveform 5



Downloaded from Arrow.com.

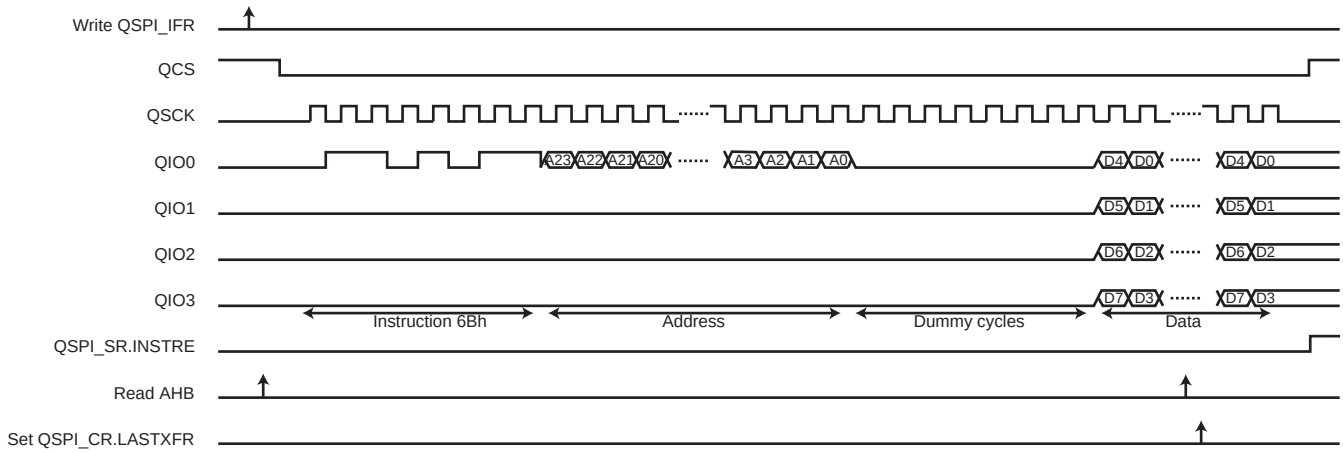
Example 6:

Instruction in Single-bit SPI, with address in Single-bit SPI, without option, with data read in Quad SPI, with eight dummy cycles.

Command: QUAD_OUTPUT READ ARRAY (6Bh)

- Write 0x0000_006B in QSPI_ICR.
- Write 0x0008_10B2 in QSPI_IFR.
- Read QSPI_IR (dummy read) to synchronize system bus accesses.
- Read data in the QSPI system bus memory space (0x8000_0000).
  The address of the first system bus read access is sent in the instruction frame.
  The address of the next system bus read accesses is not used.
- Set LASTXFR bit in QSPI_CR.
- Wait for INSTRE in QSPI_SR to rise.

Figure 41-16. Instruction Transmission Waveform 6



Downloaded from Arrow.com.

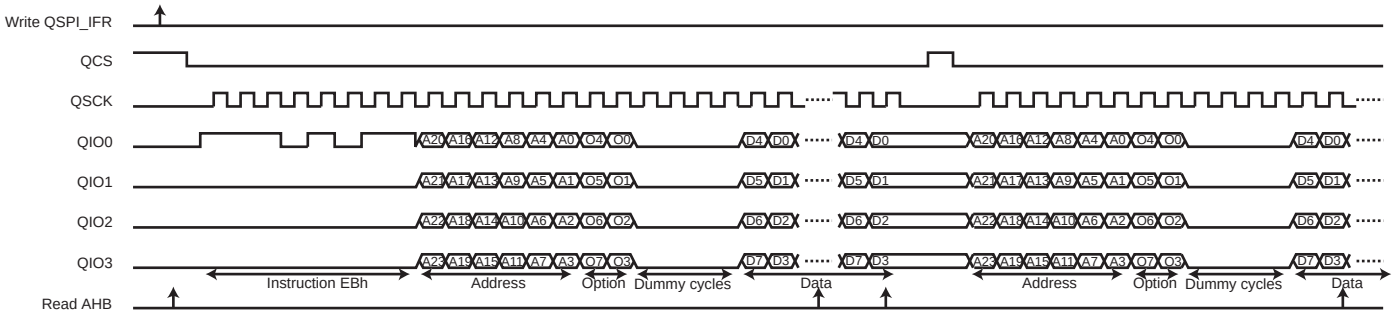
Example 7:

Instruction in Single-bit SPI, with address and option in Quad SPI, with data read in Quad SPI, with four dummy cycles, with fetch and continuous read.

Command: FAST READ QUAD I/O (EBh) - 8-BIT OPTION (0x30h)

- Write 0x0030_00EB in QSPI_ICR.
- Write 0x0004_33F4 in QSPI_IFR.
- Read QSPI_IFR (dummy read) to synchronize system bus accesses.
- Read data in the QSPI system bus memory space (0x8000_0000).
  Fetch is enabled, the address of the system bus read accesses is always used.
- Set LASTXFR bit in QSPI_CR.
- Wait for INSTRE in QSPI_SR to rise.

Figure 41-17. Instruction Transmission Waveform 7



Downloaded from Arrow.com.

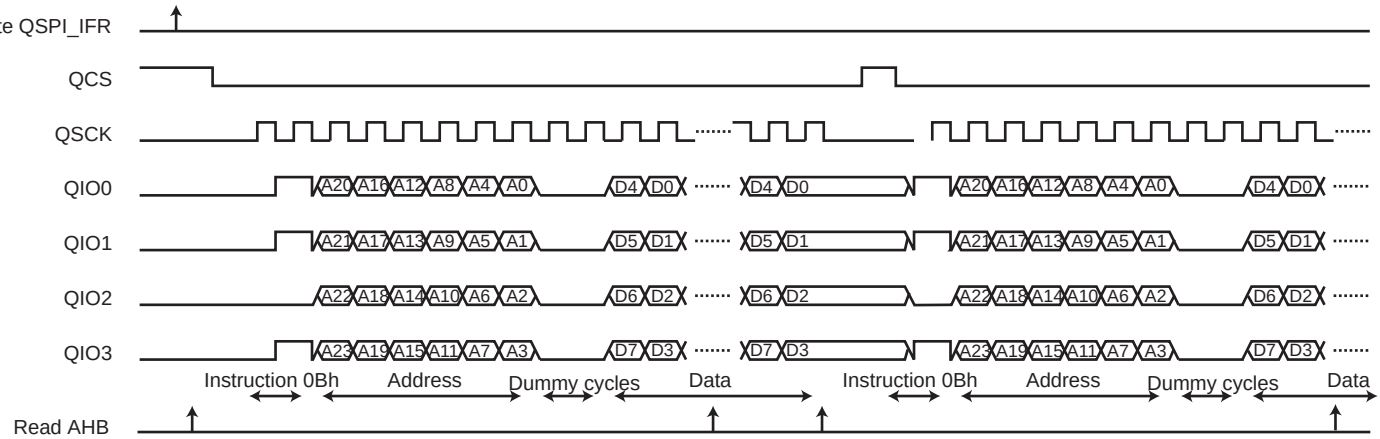
Example 8:

Instruction in Quad SPI, with address in Quad SPI, without option, with data read in Quad SPI, with two dummy cycles, with fetch.

Command: HIGH-SPEED READ (0Bh)

- Write 0x0000_000B in QSPI_ICR.
- Write 0x0002_20B6 in QSPI_IFR.
- Read QSPI_IFR (dummy read) to synchronize system bus accesses.
- Read data in the QSPI system bus memory space (0x8000_0000).
  Fetch is enabled, the address of the system bus read accesses is always used.
- Set LASTXFR bit in QSPI_CR.
- Wait for INSTRE in QSPI_SR to rise.

Figure 41-18. Instruction Transmission Waveform 8



Downloaded from Arrow.com.

41.6.6 Scrambling/Unscrambling Function

The scrambling/unscrambling function cannot be performed on devices other than memories. Data is scrambled when written to memory and unscrambled when data is read.

The external data lines can be scrambled in order to prevent intellectual property data located in off-chip memories from being easily recovered by analyzing data at the package pin level of either the microcontroller or the QSPI slave device (e.g., memory).

The scrambling/unscrambling function can be enabled by setting the SCREN bit in the QSPI Scrambling Mode Register (QSPI_SMR).

The scrambling and unscrambling are performed on-the-fly without impacting the throughput.

The scrambling method depends on the user-configurable scrambling user key (field USRK) in the QSPI Scrambling Key Register (QSPI_SKR). The QSPI_SKR is only accessible in Write mode.

If RVDIS in QSPI_SMR is cleared, the scrambling/unscrambling algorithm includes the scrambling user key plus a random value depending on device processing characteristics. Data scrambled by a given microcontroller cannot be unscrambled by another.

If bit RVDIS in QSPI_SMR is set, the scrambling/unscrambling algorithm includes only the scrambling user key. No random value is part of the key.

The scrambling user key or the seed for key generation must be securely stored in a reliable non-volatile memory in order to recover data from the off-chip memory. Any data scrambled with a given key cannot be recovered if the key is lost.

41.6.7 Register Write Protection

To prevent any single software error from corrupting QSPI behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the QSPI Write Protection Mode Register (QSPI_WPMR).

If a write access to a write-protected register is detected, the WPVS flag in the QSPI Write Protection Status Register (QSPI_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading the QSPI_WPSR.

The following registers can be write-protected:

- QSPI Mode Register
- QSPI Serial Clock Register
- QSPI Scrambling Mode Register
- QSPI Scrambling Key Register

Downloaded from Arrow.com.

41.7 Quad SPI Interface (QSPI) User Interface

Table 41-5. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	QSPI_CR	Write-only	–
0x04	Mode Register	QSPI_MR	Read/Write	0x0
0x08	Receive Data Register	QSPI_RDR	Read-only	0x0
0x0C	Transmit Data Register	QSPI_TDR	Write-only	–
0x10	Status Register	QSPI_SR	Read-only	0x0
0x14	Interrupt Enable Register	QSPI_IER	Write-only	–
0x18	Interrupt Disable Register	QSPI_IDR	Write-only	–
0x1C	Interrupt Mask Register	QSPI_IMR	Read-only	0x0
0x20	Serial Clock Register	QSPI_SCR	Read/Write	0x0
0x30	Instruction Address Register	QSPI_IAR	Read/Write	0x0
0x34	Instruction Code Register	QSPI_ICR	Read/Write	0x0
0x38	Instruction Frame Register	QSPI_IFR	Read/Write	0x0
0x3C	Reserved	–	–	–
0x40	Scrambling Mode Register	QSPI_SMR	Read/Write	0x0
0x44	Scrambling Key Register	QSPI_SKR	Write-only	–
0x48–0xE0	Reserved	–	–	–
0xE4	Write Protection Mode Register	QSPI_WPMR	Read/Write	0x0
0xE8	Write Protection Status Register	QSPI_WPSR	Read-only	0x0
0xEC–0xF8	Reserved	–	–	–
0x00FC	Reserved	–	–	–

Downloaded from Arrow.com.

41.7.1 QSPI Control Register

Name: QSPI_CR

Address: 0x4007C000

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	LASTXFER
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
SWRST	–	–	–	–	–	QSPIDIS	QSPIEN

- **QSPIEN: QSPI Enable**

0: No effect.

1: Enables the QSPI to transfer and receive data.

- **QSPIDIS: QSPI Disable**

0: No effect.

1: Disables the QSPI.

As soon as QSPIDIS is set, the QSPI finishes its transfer.

All pins are set in input mode and no data is received or transmitted.

If a transfer is in progress, the transfer is finished before the QSPI is disabled.

If both QSPIEN and QSPIDIS are equal to one when the control register is written, the QSPI is disabled.

- **SWRST: QSPI Software Reset**

0: No effect.

1: Reset the QSPI. A software-triggered hardware reset of the QSPI interface is performed.

DMA channels are not affected by software reset.

- **LASTXFER: Last Transfer**

0: No effect.

1: The chip select is deasserted after the character written in QSPI_TDR.TD has been transferred.

Downloaded from Arrow.com.

41.7.2 QSPI Mode Register

Name: QSPI_MR

Address: 0x4007C004

Access: Read/Write

31	30	29	28	27	26	25	24
DLYCS							
23	22	21	20	19	18	17	16
DLYBCT							
15	14	13	12	11	10	9	8
–	–	–	–	NBBITS			
7	6	5	4	3	2	1	0
–	–	CSMODE		–	WDRBT	LLB	SMM

This register can only be written if bit WPEN is cleared in the QSPI Write Protection Mode Register.

- **SMM: Serial Memory Mode**

0 (SPI): The QSPI is in SPI mode.

1 (MEMORY): The QSPI is in Serial Memory mode.

- **LLB: Local Loopback Enable**

0 (DISABLED): Local loopback path disabled.

1 (ENABLED): Local loopback path enabled.

LLB controls the local loopback on the data serializer for testing in SPI mode only. (MISO is internally connected on MOSI).

- **WDRBT: Wait Data Read Before Transfer**

0 (DISABLED): No effect. In SPI mode, a transfer can be initiated whatever the state of the QSPI_RDR is.

1 (ENABLED): In SPI mode, a transfer can start only if the QSPI_RDR is empty, i.e., does not contain any unread data. This mode prevents overrun error in reception.

- **CSMODE: Chip Select Mode**

The CSMODE field determines how the chip select is de-asserted.

Value	Name	Description
0	NOT_RELOADED	The chip select is deasserted if QSPI_TDR.TD has not been reloaded before the end of the current transfer.
1	LASTXFER	The chip select is deasserted when the bit LASTXFER is written at 1 and the character written in QSPI_TDR.TD has been transferred.
2	SYSTEMATICALLY	The chip select is deasserted systematically after each transfer.

Downloaded from Arrow.com.

- **NBBITS: Number Of Bits Per Transfer**

Value	Name	Description
0	8_BIT	8 bits for transfer
8	16_BIT	16 bits for transfer

- **DLYCS: Minimum Inactive QCS Delay**

This field defines the minimum delay between the deactivation and the activation of QCS. The DLYCS time guarantees the slave minimum deselect time.

If DLYCS is equal to zero, one peripheral clock period is inserted by default.

Otherwise, the following equation determines the delay:

$$\text{Minimum Inactive NPCS} = \frac{\text{DLYCS}}{f_{\text{peripheral clock}}}$$

- **DLYBCT: Delay Between Consecutive Transfers**

This field defines the delay between two consecutive transfers with the same peripheral without removing the chip select. The delay is always inserted after each transfer and before removing the chip select if needed.

When DLYBCT equals zero, no delay between consecutive transfers is inserted and the clock keeps its duty cycle over the character transfers. In Serial Memory mode (SMM bit = 1), DLYBCT is ignored and no delay is inserted.

Otherwise, the following equation determines the delay:

$$\text{Delay Between Consecutive Transfers} = \frac{32 \times \text{DLYBCT}}{f_{\text{peripheral clock}}}$$

Downloaded from Arrow.com.

41.7.3 QSPI Receive Data Register

Name: QSPI_RDR

Address: 0x4007C008

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
RD							
7	6	5	4	3	2	1	0
RD							

- **RD: Receive Data**

Data received by the QSPI is stored in this register right-justified. Unused bits read zero.

Downloaded from Arrow.com.

41.7.4 QSPI Transmit Data Register

Name: QSPI_TDR

Address: 0x4007C00C

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
TD							
7	6	5	4	3	2	1	0
TD							

- **TD: Transmit Data**

Data to be transmitted by the QSPI is stored in this register. Information to be transmitted must be written to the transmit data register in a right-justified format.

Downloaded from Arrow.com.

41.7.5 QSPI Status Register

Name: QSPI_SR

Address: 0x4007C010

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	QSPIENS
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	INSTRE	CSS	CSR
7	6	5	4	3	2	1	0
–	–	–	–	OVRES	TXEMPTY	TDRE	RDRF

- **RDRF: Receive Data Register Full (cleared by reading SPI_RDR)**

0: No data has been received since the last read of QSPI_RDR

1: Data has been received and the received data has been transferred from the serializer to QSPI_RDR since the last read of QSPI_RDR.

- **TDRE: Transmit Data Register Empty (cleared by writing SPI_TDR)**

0: Data has been written to QSPI_TDR and not yet transferred to the serializer.

1: The last data written in the QSPI_TDR has been transferred to the serializer.

TDRE equals zero when the QSPI is disabled or at reset. The QSPI enable command sets this bit to one.

- **TXEMPTY: Transmission Registers Empty (cleared by writing SPI_TDR)**

0: As soon as data is written in QSPI_TDR.

1: QSPI_TDR and internal shifter are empty. If a transfer delay has been defined, TXEMPTY is set after the completion of such delay.

- **OVRES: Overrun Error Status (cleared on read)**

0: No overrun has been detected since the last read of QSPI_SR.

1: At least one overrun error has occurred since the last read of QSPI_SR.

An overrun occurs when QSPI_RDR is loaded at least twice from the serializer since the last read of the QSPI_RDR.

- **CSR: Chip Select Rise**

0: No chip select rise has been detected since the last read of QSPI_SR.

1: At least one chip select rise has been detected since the last read of QSPI_SR.

- **CSS: Chip Select Status**

0: The chip select is asserted.

1: The chip select is not asserted.

- **INSTRE: Instruction End Status**

0: No instruction end has been detected since the last read of QSPI_SR.

1: At least one instruction end has been detected since the last read of QSPI_SR.

Downloaded from Arrow.com.

- **QSPIENS: QSPI Enable Status**

0: QSPI is disabled.

1: QSPI is enabled.

Downloaded from Arrow.com.

41.7.6 QSPI Interrupt Enable Register

Name: QSPI_IER

Address: 0x4007C014

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	INSTRE	CSS	CSR
7	6	5	4	3	2	1	0
–	–	–	–	OVRES	TXEMPTY	TDRE	RDRF

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Enables the corresponding interrupt

- **RDRF: Receive Data Register Full Interrupt Enable**
- **TDRE: Transmit Data Register Empty Interrupt Enable**
- **TXEMPTY: Transmission Registers Empty Enable**
- **OVRES: Overrun Error Interrupt Enable**
- **CSR: Chip Select Rise Interrupt Enable**
- **CSS: Chip Select Status Interrupt Enable**
- **INSTRE: Instruction End Interrupt Enable**

Downloaded from Arrow.com.

41.7.7 QSPI Interrupt Disable Register

Name: QSPI_IDR

Address: 0x4007C018

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	INSTRE	CSS	CSR
7	6	5	4	3	2	1	0
–	–	–	–	OVRES	TXEMPTY	TDRE	RDRF

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Disables the corresponding interrupt

- **RDRF: Receive Data Register Full Interrupt Disable**
- **TDRE: Transmit Data Register Empty Interrupt Disable**
- **TXEMPTY: Transmission Registers Empty Disable**
- **OVRES: Overrun Error Interrupt Disable**
- **CSR: Chip Select Rise Interrupt Disable**
- **CSS: Chip Select Status Interrupt Disable**
- **INSTRE: Instruction End Interrupt Disable**

Downloaded from Arrow.com.

41.7.8 QSPI Interrupt Mask Register

Name: QSPI_IMR

Address: 0x4007C01C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	INSTRE	CSS	CSR
7	6	5	4	3	2	1	0
–	–	–	–	OVRES	TXEMPTY	TDRE	RDRF

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

- **RDRF: Receive Data Register Full Interrupt Mask**
- **TDRE: Transmit Data Register Empty Interrupt Mask**
- **TXEMPTY: Transmission Registers Empty Mask**
- **OVRES: Overrun Error Interrupt Mask**
- **CSR: Chip Select Rise Interrupt Mask**
- **CSS: Chip Select Status Interrupt Mask**
- **INSTRE: Instruction End Interrupt Mask**

Downloaded from Arrow.com.

41.7.9 QSPI Serial Clock Register

Name: QSPI_SCR

Address: 0x4007C020

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
DLYBS							
15	14	13	12	11	10	9	8
SCBR							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	CPHA	CPOL

This register can only be written if bit WPEN is cleared in the QSPI Write Protection Mode Register.

- **CPOL: Clock Polarity**

0: The inactive state value of QSCK is logic level zero.

1: The inactive state value of QSCK is logic level one.

CPOL is used to determine the inactive state value of the serial clock (QSCK). It is used with CPHA to produce the required clock/data relationship between master and slave devices.

- **CPHA: Clock Phase**

0: Data is captured on the leading edge of QSCK and changed on the following edge of QSCK.

1: Data is changed on the leading edge of QSCK and captured on the following edge of QSCK.

CPHA determines which edge of QSCK causes data to change and which edge causes data to be captured. CPHA is used with CPOL to produce the required clock/data relationship between master and slave devices.

- **SCBR: Serial Clock Baud Rate**

The QSPI uses a modulus counter to derive the QSCK baud rate from the peripheral clock. The baud rate is selected by writing a value from 0 to 255 in the SCBR field. The following equation determines the QSCK baud rate:

$$\text{QSCK Baudrate} = \frac{f_{\text{peripheral clock}}}{(\text{SCBR} + 1)}$$

- **DLYBS: Delay Before QSCK**

This field defines the delay from QCS valid to the first valid QSCK transition.

When DLYBS equals zero, the QCS valid to QSCK transition is 1/2 the QSCK clock period.

Otherwise, the following equation determines the delay:

$$\text{Delay Before QSCK} = \frac{\text{DLYBS}}{f_{\text{peripheral clock}}}$$

Downloaded from Arrow.com.

41.7.10 QSPI Instruction Address Register

Name: QSPI_IAR

Address: 0x4007C030

Access: Read/Write

31	30	29	28	27	26	25	24
ADDR							
23	22	21	20	19	18	17	16
ADDR							
15	14	13	12	11	10	9	8
ADDR							
7	6	5	4	3	2	1	0
ADDR							

- **ADDR: Address**

Address to send to the serial Flash memory in the instruction frame.

Downloaded from Arrow.com.

41.7.11 QSPI Instruction Code Register

Name: QSPI_ICR

Address: 0x4007C034

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
OPT							

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
INST							

- **INST: Instruction Code**

Instruction code to send to the serial Flash memory.

- **OPT: Option Code**

Option code to send to the serial Flash memory.

Downloaded from Arrow.com.

41.7.12 QSPI Instruction Frame Register

Name: QSPI_IFR

Address: 0x4007C038

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	NBDUM				
15	14	13	12	11	10	9	8
–	CRM	TFRTYP		–	ADDRL	OPTL	
7	6	5	4	3	2	1	0
DATAEN	OPTEN	ADDREN	INSTEN	–	WIDTH		

- **WIDTH: Width of Instruction Code, Address, Option Code and Data**

Value	Name	Description
0	SINGLE_BIT_SPI	Instruction: Single-bit SPI / Address-Option: Single-bit SPI / Data: Single-bit SPI
1	DUAL_OUTPUT	Instruction: Single-bit SPI / Address-Option: Single-bit SPI / Data: Dual SPI
2	QUAD_OUTPUT	Instruction: Single-bit SPI / Address-Option: Single-bit SPI / Data: Quad SPI
3	DUAL_IO	Instruction: Single-bit SPI / Address-Option: Dual SPI / Data: Dual SPI
4	QUAD_IO	Instruction: Single-bit SPI / Address-Option: Quad SPI / Data: Quad SPI
5	DUAL_CMD	Instruction: Dual SPI / Address-Option: Dual SPI / Data: Dual SPI
6	QUAD_CMD	Instruction: Quad SPI / Address-Option: Quad SPI / Data: Quad SPI

- **INSTEN: Instruction Enable**

0: The instruction is not sent to the serial Flash memory.

1: The instruction is sent to the serial Flash memory.

- **ADDREN: Address Enable**

0: The transfer address is not sent to the serial Flash memory.

1: The transfer address is sent to the serial Flash memory.

- **OPTEN: Option Enable**

0: The option is not sent to the serial Flash memory.

1: The option is sent to the serial Flash memory.

- **DATAEN: Data Enable**

0: No data is sent/received to/from the serial Flash memory.

1: Data is sent/received to/from the serial Flash memory.

Downloaded from Arrow.com.

- **OPTL: Option Code Length**

The OPTL field determines the length of the option code. The value written in OPTL must be coherent with value written in the field WIDTH. For example: OPTL = 0 (1-bit option code) is not coherent with WIDTH = 6 (option code sent with Quad-SPI protocol, thus the minimum length of the option code is 4-bit).

Value	Name	Description
0	OPTION_1BIT	The option code is 1 bit long.
1	OPTION_2BIT	The option code is 2 bits long.
2	OPTION_4BIT	The option code is 4 bits long.
3	OPTION_8BIT	The option code is 8 bits long.

- **ADDRL: Address Length**

The ADDRL bit determines the length of the address.

0 (24_BIT): The address is 24 bits long.

1 (32_BIT): The address is 32 bits long.

- **TFRTYP: Data Transfer Type**

Value	Name	Description
0	TRSFR_READ	Read transfer from the serial memory. Scrambling is not performed. Read at random location (fetch) in the serial Flash memory is not possible.
1	TRSFR_READ_MEMORY	Read data transfer from the serial memory. If enabled, scrambling is performed. Read at random location (fetch) in the serial Flash memory is possible.
2	TRSFR_WRITE	Write transfer into the serial memory. Scrambling is not performed.
3	TRSFR_WRITE_MEMORY	Write data transfer into the serial memory. If enabled, scrambling is performed.

- **CRM: Continuous Read Mode**

0 (DISABLED): The Continuous Read mode is disabled.

1 (ENABLED): The Continuous Read mode is enabled.

- **NBDUM: Number Of Dummy Cycles**

The NBDUM field defines the number of dummy cycles required by the serial Flash memory before data transfer.

Downloaded from Arrow.com.

41.7.13 QSPI Scrambling Mode Register

Name: QSPI_SMR

Address: 0x4007C040

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	RVDIS	SCREN

This register can only be written if bit WPEN is cleared in the QSPI Write Protection Mode Register.

- **SCREN: Scrambling/Unscrambling Enable**

0 (DISABLED): The scrambling/unscrambling is disabled.

1 (ENABLED): The scrambling/unscrambling is enabled.

- **RVDIS: Scrambling/Unscrambling Random Value Disable**

0: The scrambling/unscrambling algorithm includes the scrambling user key plus a random value that may differ from chip to chip.

1: The scrambling/unscrambling algorithm includes only the scrambling user key.

Downloaded from Arrow.com.

41.7.14 QSPI Scrambling Key Register

Name: QSPI_SKR

Address: 0x4007C044

Access: Write-only

31	30	29	28	27	26	25	24
USRK							
23	22	21	20	19	18	17	16
USRK							
15	14	13	12	11	10	9	8
USRK							
7	6	5	4	3	2	1	0
USRK							

This register can only be written if bit WPEN is cleared in the QSPI Write Protection Mode Register.

- **USRK: Scrambling User Key**

Downloaded from Arrow.com.

41.7.15 QSPI Write Protection Mode Register

Name: QSPI_WPMR

Address: 0x4007C0E4

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							
23	22	21	20	19	18	17	16
WPKEY							
15	14	13	12	11	10	9	8
WPKEY							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x515350 (QSP in ASCII)

1: Enables the write protection if WPKEY corresponds to 0x515350 (QSP in ASCII)

See Section 41.6.7 ”Register Write Protection” for the list of registers that can be protected.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x515350	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

Downloaded from Arrow.com.

41.7.16 QSPI Write Protection Status Register

Name: QSPI_WPSR

Address: 0x4007C0E8

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
WPVSRC							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

- **WPVS: Write Protection Violation Status**

0: No write protection violation has occurred since the last read of the QSPI_WPSR.

1: A write protection violation has occurred since the last read of the QSPI_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protection Violation Source**

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

Downloaded from Arrow.com.

42. Two-wire Interface (TWIHS)

42.1 Description

The Atmel Two-wire Interface (TWIHS) interconnects components on a unique two-wire bus, made up of one clock line and one data line with speeds of up to 400 kbit/s in Fast mode and up to 3.4 Mbit/s in High-speed slave mode only, based on a byte-oriented transfer format. It can be used with any Atmel Two-wire Interface bus Serial EEPROM and I²C-compatible devices, such as a Real-Time Clock (RTC), Dot Matrix/Graphic LCD Controller and temperature sensor. The TWIHS is programmable as a master or a slave with sequential or single-byte access. Multiple master capability is supported.

A configurable baud rate generator permits the output data rate to be adapted to a wide range of core clock frequencies.

Table 42-1 lists the compatibility level of the Atmel Two-wire Interface in Master mode and a full I²C compatible device.

Table 42-1. Atmel TWI Compatibility with I²C Standard

I²C Standard	Atmel TWI
Standard Mode Speed (100 kHz)	Supported
Fast Mode Speed (400 kHz)	Supported
High-speed Mode (Slave only, 3.4 MHz)	Supported
7- or 10-bit(1) Slave Addressing	Supported
START Byte(2)	Not Supported
Repeated Start (Sr) Condition	Supported
ACK and NACK Management	Supported
Input Filtering	Supported
Slope Control	Not Supported
Clock Stretching	Supported
Multi Master Capability	Supported

Notes: 1. 10-bit support in Master mode only
2. START + b000000001 + Ack + Sr

42.2 Embedded Characteristics

- 3 TWIHSs
- Compatible with Atmel Two-wire Interface Serial Memory and I²C Compatible Devices(1)
- One, Two or Three Bytes for Slave Address
- Sequential Read/Write Operations
- Master and Multi-master Operation (Standard and Fast Modes Only)
- Slave Mode Operation (Standard, Fast and High-Speed Modes)
- Bit Rate: Up to 400 Kbit/s in Fast Mode and 3.4 Mbit/s in High-Speed Mode (Slave Mode Only)
- General Call Supported in Slave Mode
- SleepWalking (Asynchronous and Partial Wake-up)
- SMBus Support
- Connection to DMA Controller (DMA) Channel Capabilities Optimizes Data Transfers
- Register Write Protection

Note: 1. See Table 42-1 for details on compatibility with I²C Standard.

Downloaded from Arrow.com.

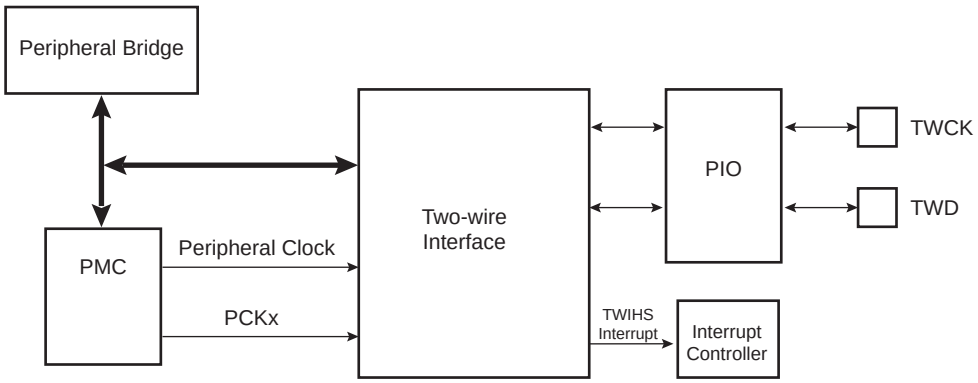
42.3 List of Abbreviations

Table 42-2. Abbreviations

Abbreviation	Description
TWI	Two-wire Interface
A	Acknowledge
NA	Non Acknowledge
P	Stop
S	Start
Sr	Repeated Start
SADR	Slave Address
ADR	Any address except SADR
R	Read
W	Write

42.4 Block Diagram

Figure 42-1. Block Diagram



42.4.1 I/O Lines Description

Table 42-3. I/O Lines Description

Pin Name	Pin Description	Type
TWD	Two-wire Serial Data	Input/Output
TWCK	Two-wire Serial Clock	Input/Output

Downloaded from Arrow.com.

42.5 Product Dependencies

42.5.1 I/O Lines

Both TWD and TWCK are bidirectional lines, connected to a positive supply voltage via a current source or pull-up resistor. When the bus is free, both lines are high. The output stages of devices connected to the bus must have an open-drain or open-collector to perform the wired-AND function.

TWD and TWCK pins may be multiplexed with PIO lines. To enable the TWIHS, the user must program the PIO Controller to dedicate TWD and TWCK as peripheral lines. When High-speed Slave mode is enabled, the analog pad filter must be enabled.

The user must not program TWD and TWCK as open-drain. This is already done by the hardware.

Table 42-4. I/O Lines

Instance	Signal	I/O Line	Peripheral
TWIHS0	TWCK0	PA4	A
TWIHS0	TWD0	PA3	A
TWIHS1	TWCK1	PB5	A
TWIHS1	TWD1	PB4	A
TWIHS2	TWCK2	PD28	C
TWIHS2	TWD2	PD27	C

42.5.2 Power Management

Enable the peripheral clock.

The TWIHS may be clocked through the Power Management Controller (PMC), thus the user must first configure the PMC to enable the TWIHS clock.

42.5.3 Interrupt Sources

The TWIHS has an interrupt line connected to the Interrupt Controller. In order to handle interrupts, the Interrupt Controller must be programmed before configuring the TWIHS. .

Table 42-5. Peripheral IDs

Instance	ID
TWIHS0	19
TWIHS1	20
TWIHS2	41

Downloaded from Arrow.com.

42.6 Functional Description

42.6.1 Transfer Format

The data put on the TWD line must be 8 bits long. Data is transferred MSB first; each byte must be followed by an acknowledgement. The number of bytes per transfer is unlimited. See Figure 42-3.

Each transfer begins with a START condition and terminates with a STOP condition. See Figure 42-2.

- A high-to-low transition on the TWD line while TWCK is high defines the START condition.
- A low-to-high transition on the TWD line while TWCK is high defines the STOP condition.

Figure 42-2. START and STOP Conditions

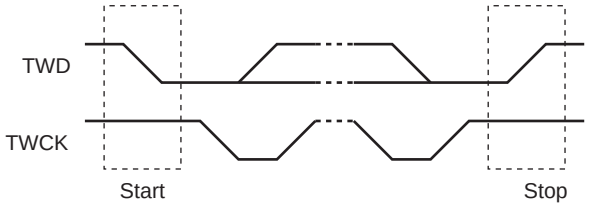
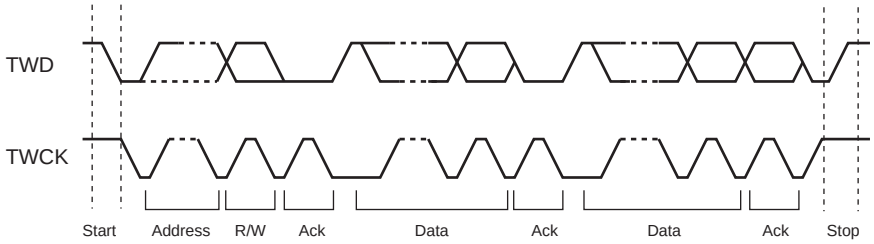


Figure 42-3. Transfer Format



42.6.2 Modes of Operation

The TWIHS has different modes of operation:

- Master Transmitter mode (Standard and Fast modes only)
- Master Receiver mode (Standard and Fast modes only)
- Multi-master Transmitter mode (Standard and Fast modes only)
- Multi-master Receiver mode (Standard and Fast modes only)
- Slave Transmitter mode (Standard, Fast and High-speed modes)
- Slave Receiver mode (Standard, Fast and High-speed modes)

These modes are described in the following sections.

Downloaded from Arrow.com.

42.6.3 Master Mode

42.6.3.1 Definition

The master is the device that starts a transfer, generates a clock and stops it. This operating mode is not available if High-speed mode is selected.

42.6.3.2 Programming Master Mode

The following registers must be programmed before entering Master mode:

1. TWIHS_MMR.DADR (+ IADRSZ + IADR if a 10-bit device is addressed): The device address is used to access slave devices in Read or Write mode.
2. TWIHS_CWGR.CKDIV + CHDIV + CLDIV: Clock Waveform register
3. TWIHS_CR.SVDIS: Disables the Slave mode
4. TWIHS_CR.MSEN: Enables the Master mode

Note: If the TWIHS is already in Master mode, the device address (DADR) can be configured without disabling the Master mode.

42.6.3.3 Transfer Rate Clock Source

The TWIHS speed is defined in the TWIHS_CWGR. The TWIHS baud rate can be based either on the peripheral clock if the CKSRC bit value is ‘0’ or on a PMC PCKx clock if the CKSRC bit value is ‘1’.

If CKSRC = 1, the baud rate is independent of the system/core clock (MCK) and thus the MCK frequency can be changed without affecting the TWIHS transfer rate.

The PMC PCKx frequency must always be three times lower than the peripheral clock frequency.

42.6.3.4 Master Transmitter Mode

This operating mode is not available if High-speed mode is selected.

After the master initiates a START condition when writing into the Transmit Holding register (TWIHS_THR), it sends a 7-bit slave address, configured in the Master Mode register (DADR in TWIHS_MMR), to notify the slave device. The bit following the slave address indicates the transfer direction, 0 in this case (MREAD = 0 in TWIHS_MMR).

The TWIHS transfers require the slave to acknowledge each received byte. During the acknowledge clock pulse (9th pulse), the master releases the data line (HIGH), enabling the slave to pull it down in order to generate the acknowledge. If the slave does not acknowledge the byte, then the Not Acknowledge flag (NACK) is set in the TWIHS Status Register (TWIHS_SR) of the master and a STOP condition is sent. The NACK flag must be cleared by reading TWIHS_SR before the next write into TWIHS_THR. As with the other status bits, an interrupt can be generated if enabled in the Interrupt Enable register (TWIHS_IER). If the slave acknowledges the byte, the data written in the TWIHS_THR is then shifted in the internal shifter and transferred. When an acknowledge is detected, the TXRDY bit is set until a new write in the TWIHS_THR.

TXRDY is used as Transmit Ready for the DMA transmit channel.

While no new data is written in the TWIHS_THR, the serial clock line is tied low. When new data is written in the TWIHS_THR, the SCL is released and the data is sent. Setting the STOP bit in TWIHS_CR generates a STOP condition.

After a master write transfer, the serial clock line is stretched (tied low) as long as no new data is written in the TWIHS_THR or until a STOP command is performed.

To clear the TXRDY flag, first set the bit TWIHS_CR.MSDIS, then set the bit TWIHS_CR.MSEN.

See Figure 42-4, Figure 42-5, and Figure 42-6.

Downloaded from Arrow.com.

Figure 42-4. Master Write with One Data Byte

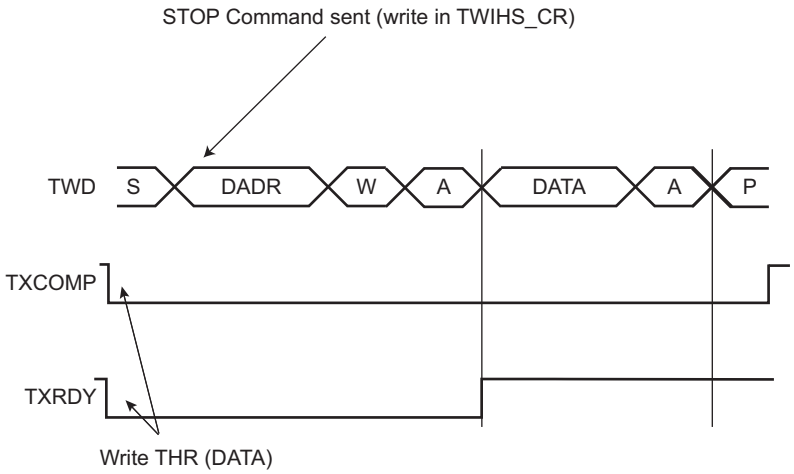


Figure 42-5. Master Write with Multiple Data Bytes

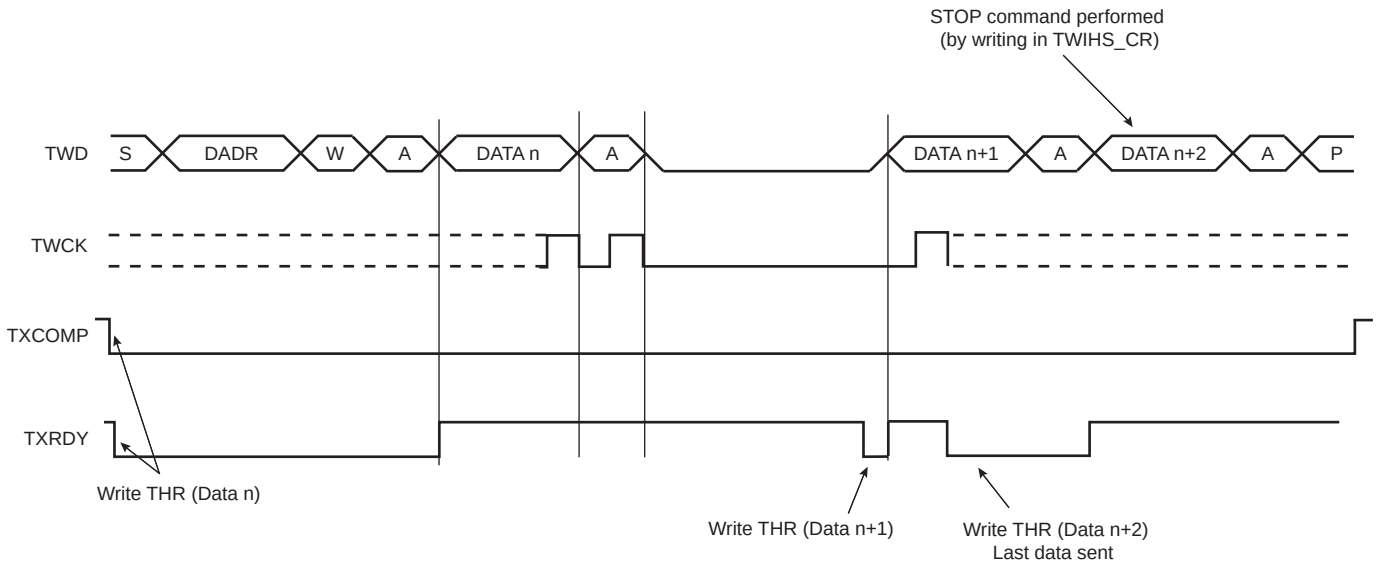
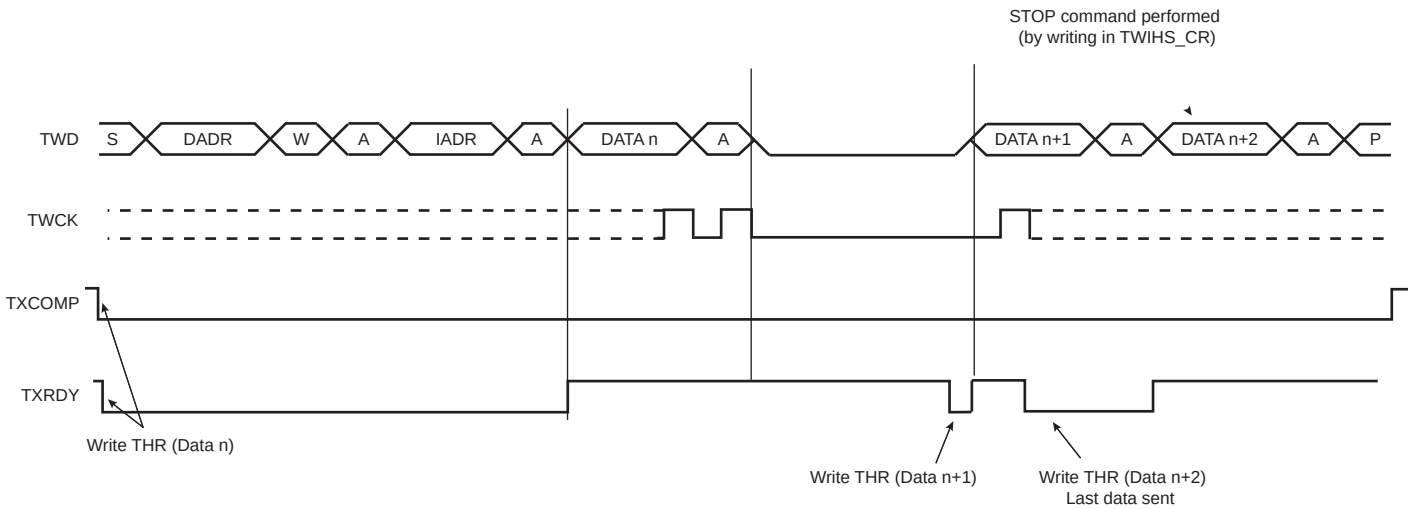


Figure 42-6. Master Write with One-Byte Internal Address and Multiple Data Bytes



Downloaded from Arrow.com.

42.6.3.5 Master Receiver Mode

Master Receiver mode is not available if High-speed mode is selected.

The read sequence begins by setting the START bit. After the START condition has been sent, the master sends a 7-bit slave address to notify the slave device. The bit following the slave address indicates the transfer direction, 1 in this case (MREAD = 1 in TWIHS_MMR). During the acknowledge clock pulse (9th pulse), the master releases the data line (HIGH), enabling the slave to pull it down in order to generate the acknowledge. The master polls the data line during this clock pulse and sets the NACK bit in the TWIHS_SR if the slave does not acknowledge the byte.

If an acknowledge is received, the master is then ready to receive data from the slave. After data has been received, the master sends an acknowledge condition to notify the slave that the data has been received except for the last data (see Figure 42-7). When the RXRDY bit is set in the TWIHS_SR, a character has been received in the Receive Holding register (TWIHS_RHR). The RXRDY bit is reset when reading the TWIHS_RHR.

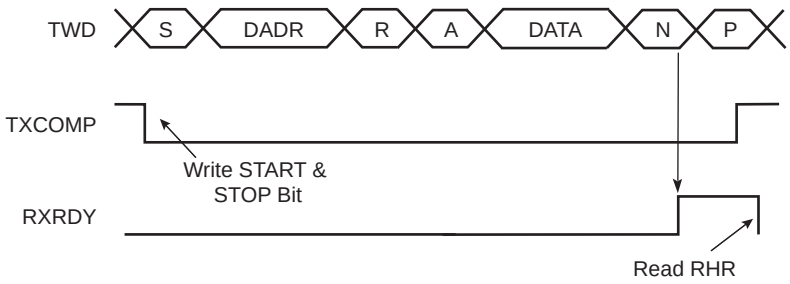
When a single data byte read is performed, with or without internal address **(**IADR), the START and STOP bits must be set at the same time. See Figure 42-7. When a multiple data byte read is performed, with or without internal address **(**IADR), the STOP bit must be set after the next-to-last data received (same condition applies for START bit to generate a REPEATED START). See Figure 42-8. For internal address usage, see Section 42.6.3.6 ”Internal Address”.

If TWIHS_RHR is full (RXRDY high) and the master is receiving data, the serial clock line is tied low before receiving the last bit of the data and until the TWIHS_RHR is read. Once the TWIHS_RHR is read, the master stops stretching the serial clock line and ends the data reception. See Figure 42-9.

Warning: When receiving multiple bytes in Master Read mode, if the next-to-last access is not read (the RXRDY flag remains high), the last access is not completed until TWIHS_RHR is read. The last access stops on the next-to-last bit (clock stretching). When the TWIHS_RHR is read, there is only half a bit period to send the STOP (or START) command, else another read access might occur (spurious access).

A possible workaround is to set the STOP (or START) bit before reading the TWIHS_RHR on the next-to-last access (within IT handler).

Figure 42-7. Master Read with One Data Byte



Downloaded from Arrow.com.

Figure 42-8. Master Read with Multiple Data Bytes

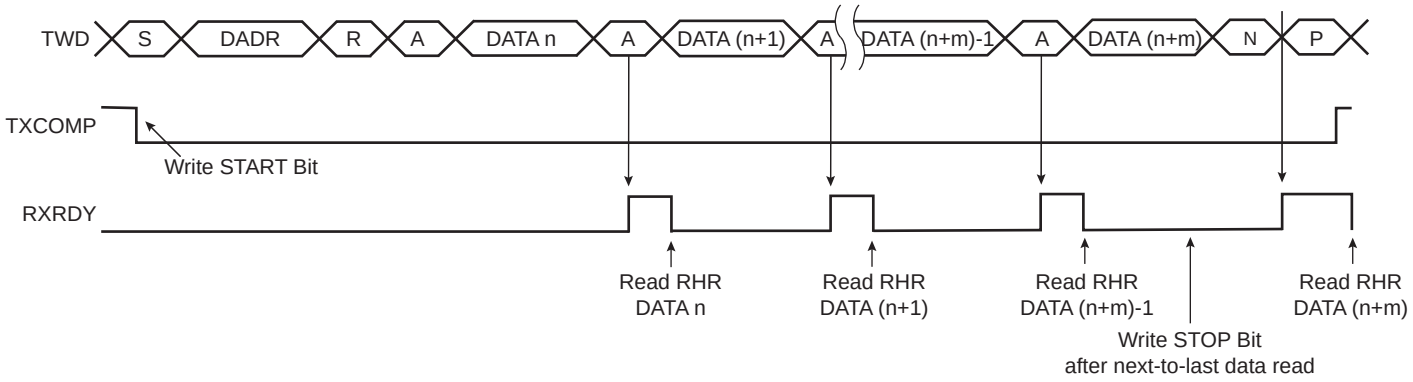
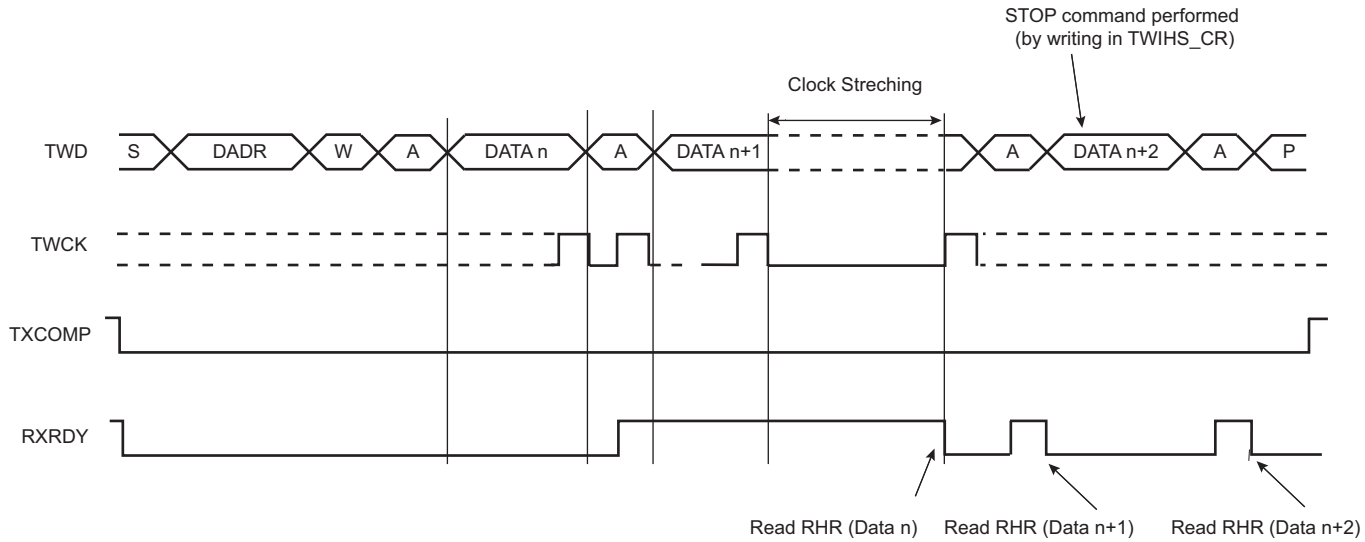


Figure 42-9. Master Read Clock Stretching with Multiple Data Bytes



RXRDY is used as receive ready for the DMA receive channel.

42.6.3.6 Internal Address

The TWIHS can perform transfers with 7-bit slave address devices and with 10-bit slave address devices.

7-bit Slave Addressing

When addressing 7-bit slave devices, the internal address bytes are used to perform random address (read or write) accesses to reach one or more data bytes, e.g. within a memory page location in a serial memory. When performing read operations with an internal address, the TWIHS performs a write operation to set the internal address into the slave device, and then switch to Master Receiver mode. Note that the second START condition (after sending the IADR) is sometimes called “repeated start” (Sr) in I^2C fully-compatible devices. See Figure 42-11.

See Figure 42-10 and Figure 42-12 for the master write operation with internal address.

The three internal address bytes are configurable through TWIHS_MMR.

If the slave device supports only a 7-bit address, i.e., no internal address, IADRSZ must be set to 0.

Downloaded from Arrow.com.

Table 42-6 shows the abbreviations used in Figure 42-10 and Figure 42-11.

Table 42-6. Abbreviations

Abbreviation	Definition
S	Start
Sr	Repeated Start
P	Stop
W	Write
R	Read
A	Acknowledge
NA	Not Acknowledge
DADR	Device Address
IADR	Internal Address

Figure 42-10. Master Write with One-, Two- or Three-Byte Internal Address and One Data Byte

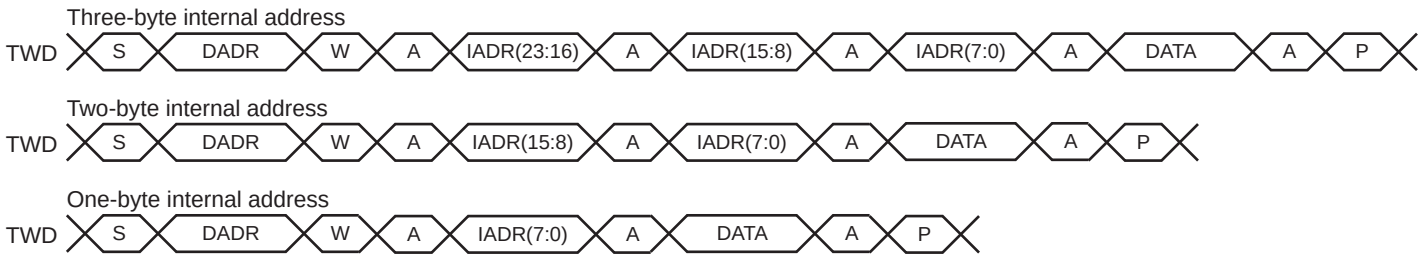
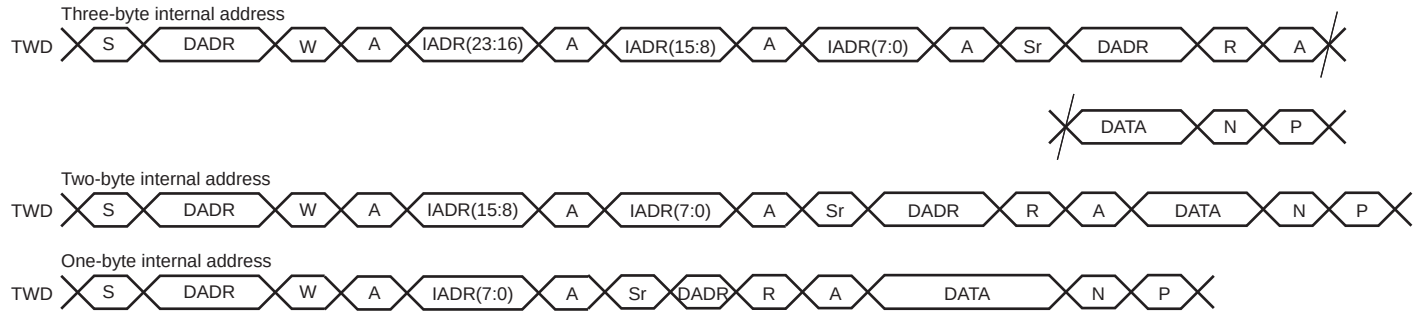


Figure 42-11. Master Read with One-, Two- or Three-Byte Internal Address and One Data Byte



10-bit Slave Addressing

For a slave address higher than seven bits, configure the address size (IADRSZ) and set the other slave address bits in the Internal Address register (TWIHS_IADR). The two remaining internal address bytes, IADR[15:8] and IADR[23:16], can be used the same way as in 7-bit slave addressing.

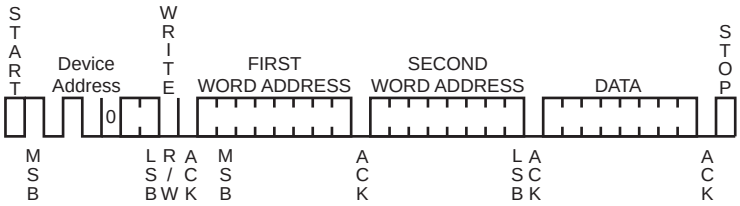
Example: Address a 10-bit device (10-bit device address is b1 b2 b3 b4 b5 b6 b7 b8 b9 b10)

1. Program IADRSZ = 1,
2. Program DADR with 1 1 1 1 0 b1 b2 (b1 is the MSB of the 10-bit address, b2, etc.)
3. Program TWIHS_IADR with b3 b4 b5 b6 b7 b8 b9 b10 (b10 is the LSB of the 10-bit address)

Figure 42-12 shows a byte write to a memory device. This demonstrates the use of internal addresses to access the device.

Downloaded from Arrow.com.

Figure 42-12. Internal Address Usage



42.6.3.7 Repeated Start

In addition to Internal Address mode, REPEATED START (Sr) can be generated manually by writing the START bit at the end of a transfer instead of the STOP bit. In such case, the parameters of the next transfer (direction, SADR, etc.) need to be set before writing the START bit at the end of the previous transfer.

See Section 42.6.3.12 "Read/Write Flowcharts" for detailed flowcharts.

Note that generating a REPEATED START after a single data read is not supported.

42.6.3.8 Bus Clear Command

The TWIHS can perform a Bus Clear command:

1. Configure the Master mode (DADR, CKDIV, etc).
2. Start the transfer by setting the CLEAR bit in the TWIHS_CR.

42.6.3.9 Using the DMA Controller (DMAC) in Master Mode

The use of the DMA significantly reduces the CPU load.

To ensure correct implementation, follow the programming sequences below:

Data Transmit with the DMA in Master Mode

The DMA transfer size must be defined with the buffer size minus 1. The remaining character must be managed without DMA to ensure that the exact number of bytes are transmitted regardless of system bus latency conditions during the end of the buffer transfer period.

1. Initialize the DMA (channels, memory pointers, size -1, etc.);
2. Configure the Master mode (DADR, CKDIV, MREAD = 0, etc.) or Slave mode.
3. Enable the DMA.
4. Wait for the DMA status flag indicating that the buffer transfer is complete.
5. Disable the DMA.
6. Wait for the TXRDY flag in TWIHS_SR.
7. Set the STOP bit in TWIHS_CR.
8. Write the last character in TWIHS_THR.
9. (Only if peripheral clock must be disabled) Wait for the TXCOMP flag to be raised in TWIHS_SR.

Data Receive with the DMA in Master Mode

The DMA transfer size must be defined with the buffer size minus 2. The two remaining characters must be managed without DMA to ensure that the exact number of bytes are received regardless of system bus latency conditions encountered during the end of buffer transfer period.

1. Initialize the DMA (channels, memory pointers, size -2, etc.);
2. Configure the Master mode (DADR, CKDIV, MREAD = 1, etc.) or Slave mode.
3. Enable the DMA.
4. (Master Only) Write the START bit in the TWIHS_CR to start the transfer.
5. Wait for the DMA status flag indicating that the buffer transfer is complete.

Downloaded from Arrow.com.

6. Disable the DMA.
7. Wait for the RXRDY flag in the TWIHS_SR.
8. Set the STOP bit in TWIHS_CR.
9. Read the penultimate character in TWIHS_RHR.
10. Wait for the RXRDY flag in the TWIHS_SR.
11. Read the last character in TWIHS_RHR.
12. (Only if peripheral clock must be disabled) Wait for the TXCOMP flag to be raised in TWIHS_SR.

42.6.3.10 SMBus Mode

SMBus mode is enabled when a one is written to the SMEN bit in the TWIHS_CR. SMBus mode operation is similar to I²C operation with the following exceptions:

1. Only 7-bit addressing can be used.
2. The SMBus standard describes a set of timeout values to ensure progress and throughput on the bus. These timeout values must be programmed into TWIHS_SMBTR.
3. Transmissions can optionally include a CRC byte, called Packet Error Check (PEC).
4. A dedicated bus line, SMBALERT, allows a slave to get a master attention.
5. A set of addresses has been reserved for protocol handling, such as alert response address (ARA) and host header (HH) address. Address matching on these addresses can be enabled by configuring the TWIHS_CR.

Packet Error Checking

Each SMBus transfer can optionally end with a CRC byte, called the PEC byte. Writing a one to the PECEN bit in TWIHS_CR enables automatic PEC handling in the current transfer. Transfers with and without PEC can be intermixed in the same system, since some slaves may not support PEC. The PEC LFSR is always updated on every bit transmitted or received, so that PEC handling on combined transfers is correct.

In Master Transmitter mode, the master calculates a PEC value and transmits it to the slave after all data bytes have been transmitted. Upon reception of this PEC byte, the slave compares it to the PEC value it has computed itself. If the values match, the data was received correctly, and the slave returns an ACK to the master. If the PEC values differ, data was corrupted, and the slave returns a NACK value. Some slaves may not be able to check the received PEC in time to return a NACK if an error occurred. In this case, the slave should always return an ACK after the PEC byte, and another method must be used to verify that the transmission was received correctly.

In Master Receiver mode, the slave calculates a PEC value and transmits it to the master after all data bytes have been transmitted. Upon reception of this PEC byte, the master compares it to the PEC value it has computed itself. If the values match, the data was received correctly. If the PEC values differ, data was corrupted, and the PECERR bit in TWIHS_SR is set. In Master Receiver mode, the PEC byte is always followed by a NACK transmitted by the master, since it is the last byte in the transfer.
In combined transfers, the PECRQ bit should only be set in the last of the combined transfers.

Consider the following transfer:

S, ADR+W, COMMAND_BYTE, ACK, SR, ADR+R, DATA_BYTE, ACK, PEC_BYTE, NACK, P

See Section 42.6.3.12 ”Read/Write Flowcharts” for detailed flowcharts.

Timeouts

The TLOWS and TLOWM fields in TWIHS_SMBTR configure the SMBus timeout values. If a timeout occurs, the master transmits a STOP condition and leaves the bus. The TOUT bit is also set in TWIHS_SR.

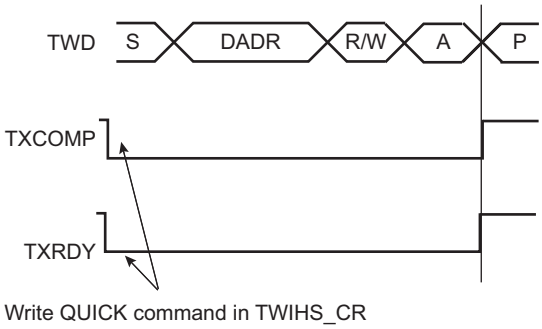
Downloaded from Arrow.com.

42.6.3.11 SMBus Quick Command (Master Mode Only)

The TWIHS can perform a quick command:

1. Configure the Master mode (DADR, CKDIV, etc).
2. Write the MREAD bit in the TWIHS_MMR at the value of the one-bit command to be sent.
3. Start the transfer by setting the QUICK bit in the TWIHS_CR.

Figure 42-13. SMBus Quick Command

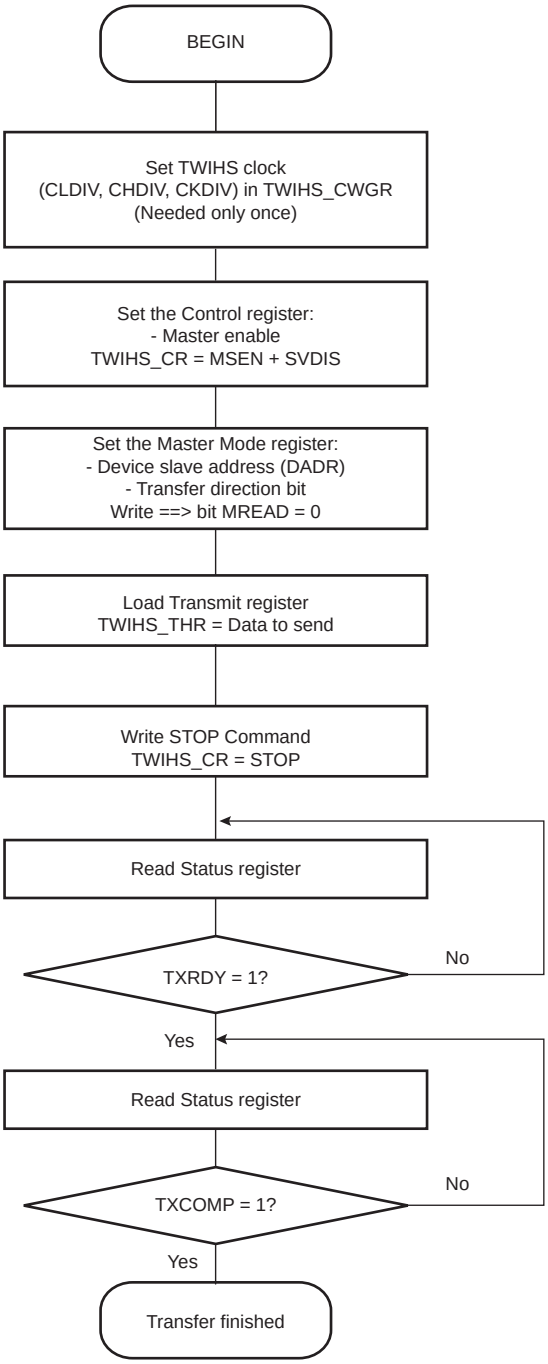


42.6.3.12 Read/Write Flowcharts

The flowcharts give examples for read and write operations. A polling or interrupt method can be used to check the status bits. The interrupt method requires that TWIHS_IER be configured first.

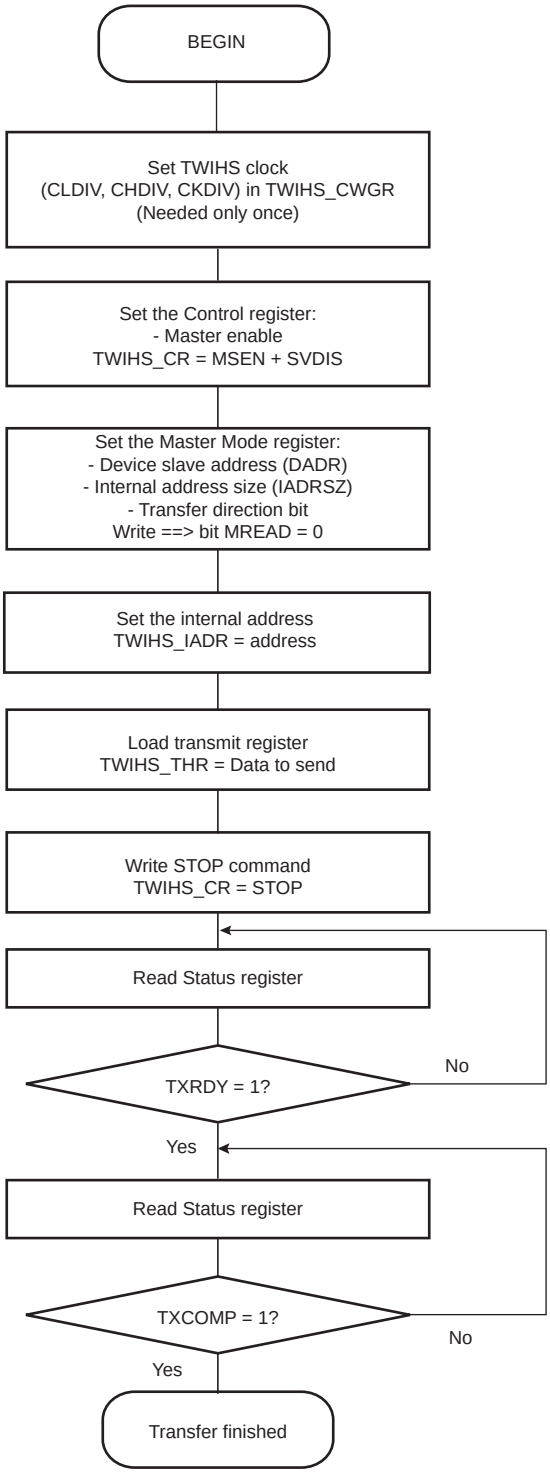
Downloaded from Arrow.com.

Figure 42-14. TWIHS Write Operation with Single Data Byte without Internal Address



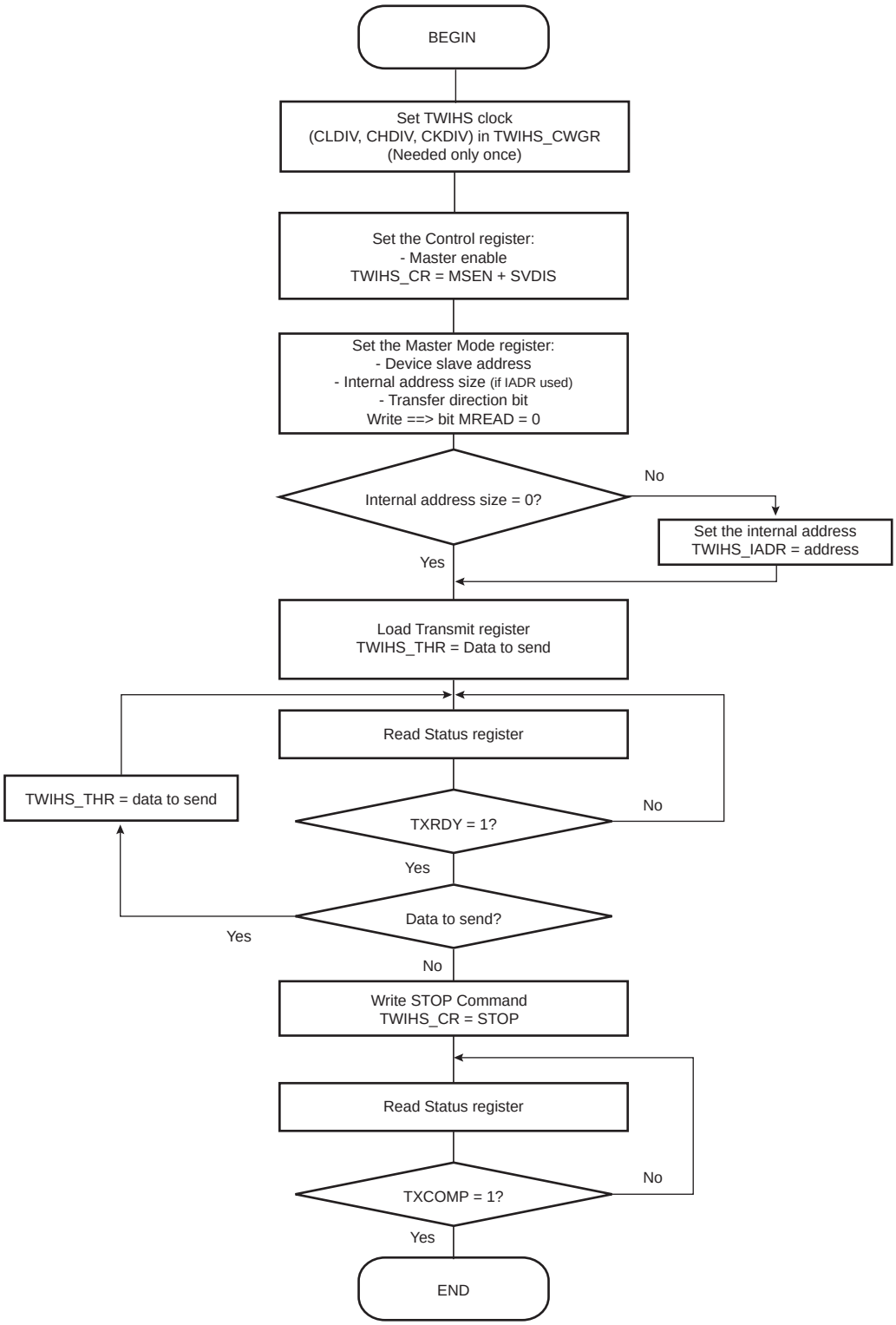
Downloaded from Arrow.com.

Figure 42-15. TWIHS Write Operation with Single Data Byte and Internal Address



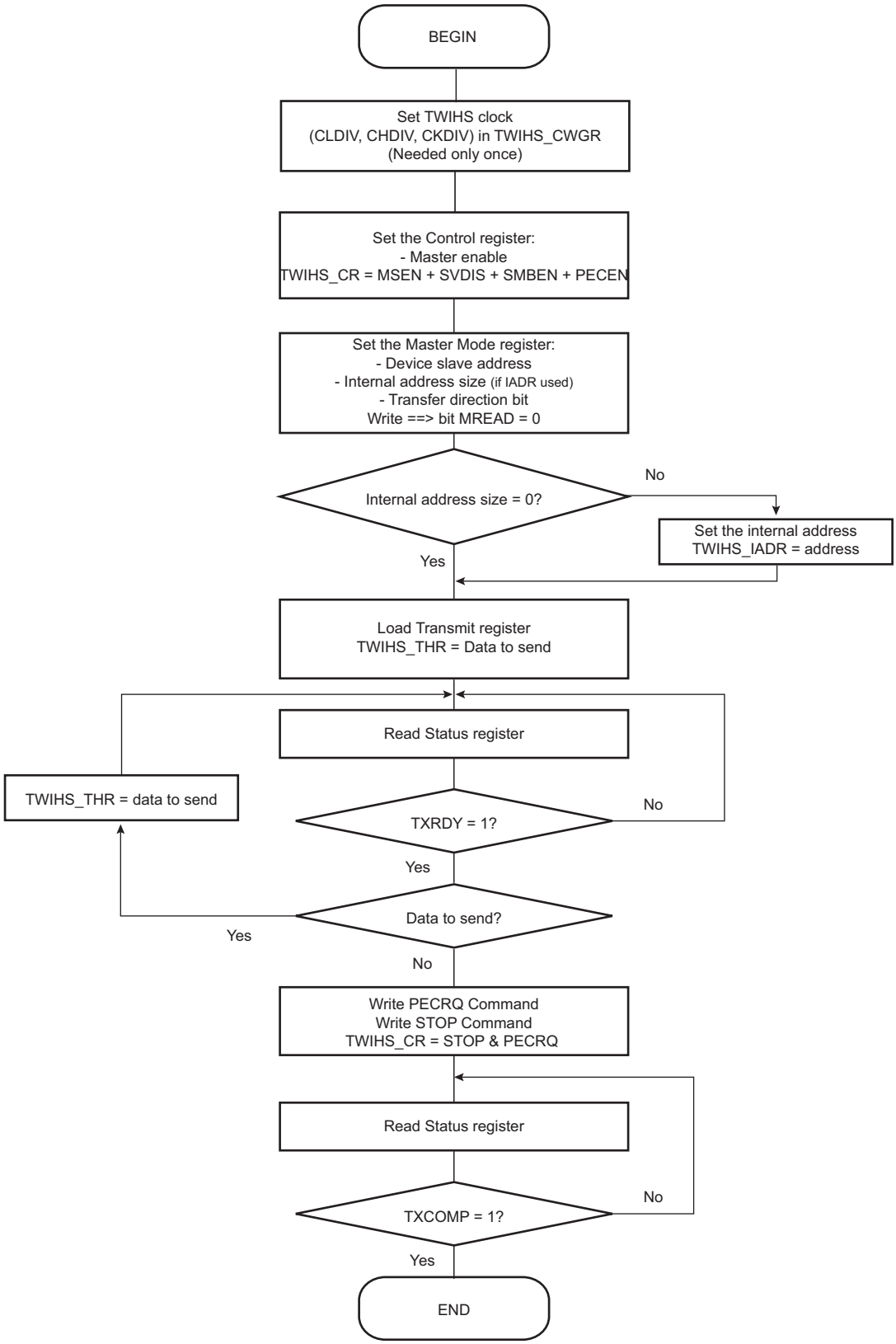
Downloaded from Arrow.com.

Figure 42-16. TWIHS Write Operation with Multiple Data Bytes with or without Internal Address



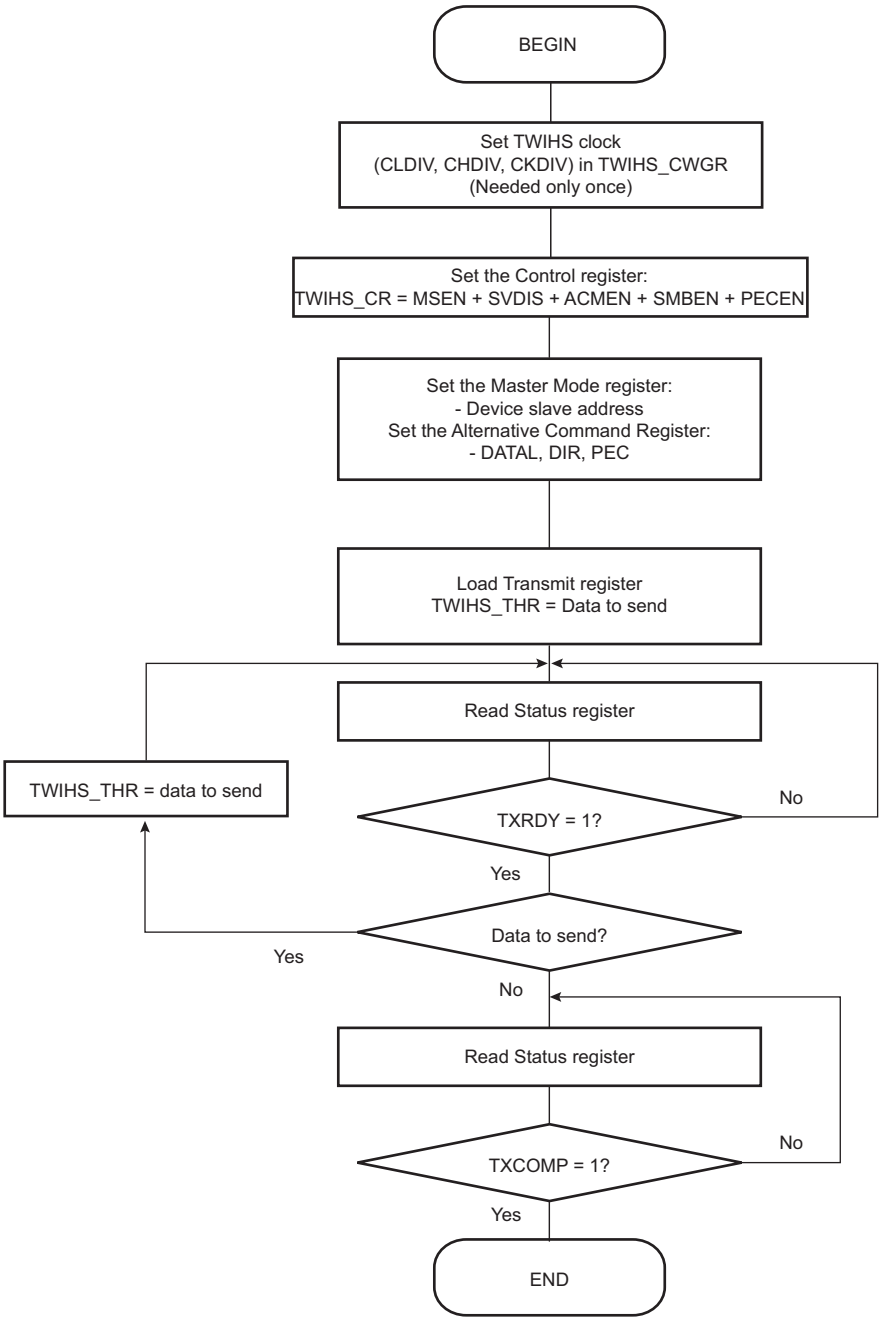
Downloaded from Arrow.com.

Figure 42-17. SMBus Write Operation with Multiple Data Bytes with or without Internal Address and PEC Sending



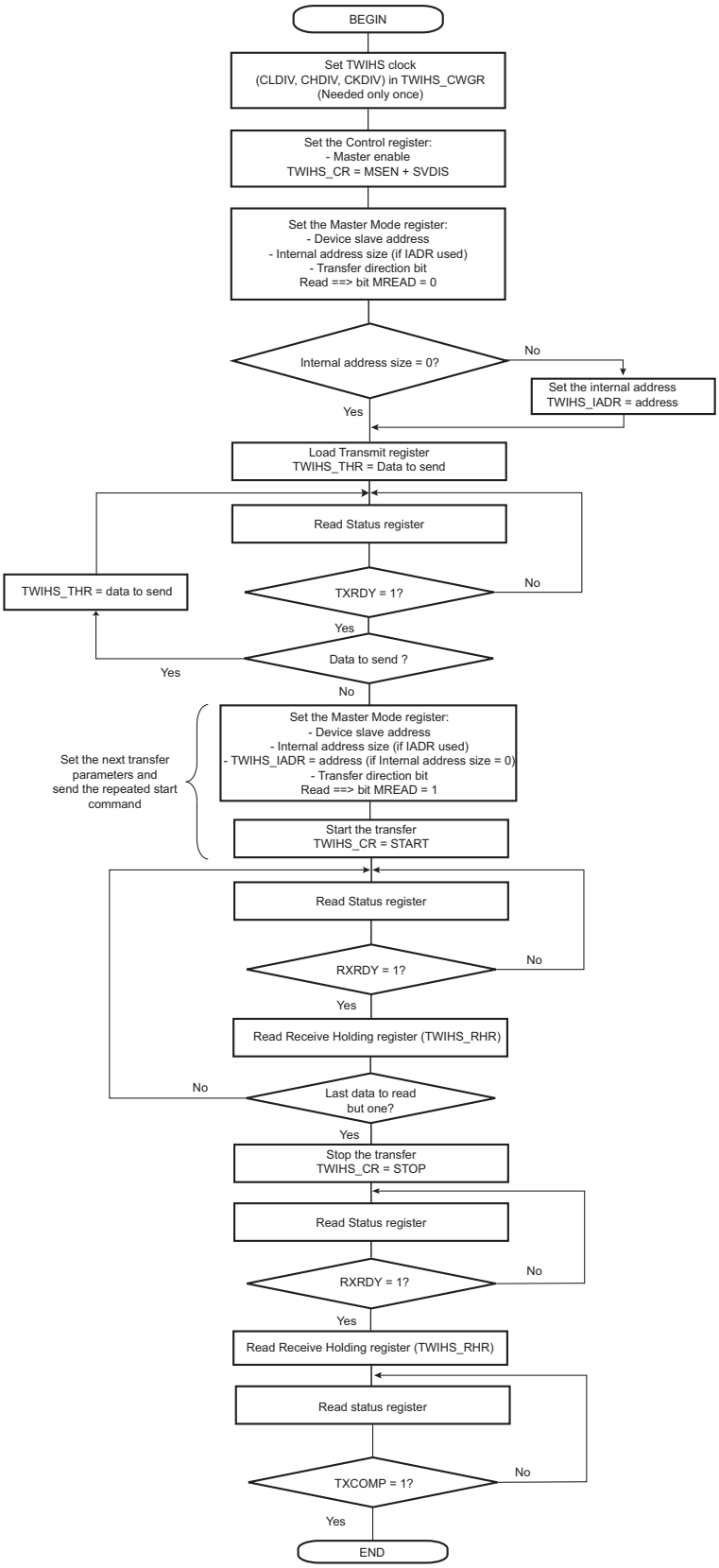
Downloaded from Arrow.com.

Figure 42-18. SMBus Write Operation with Multiple Data Bytes with PEC and Alternative Command Mode



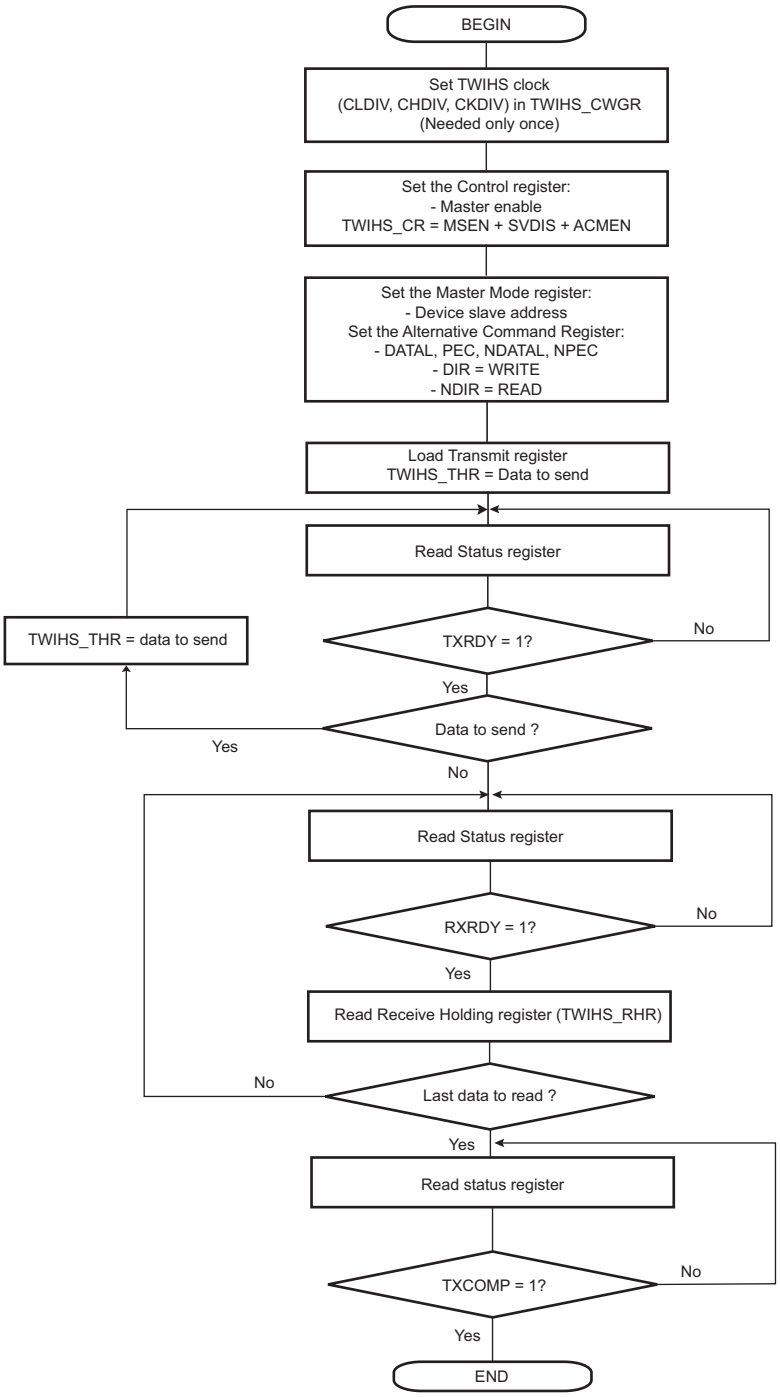
Downloaded from Arrow.com.

Figure 42-19. TWIHS Write Operation with Multiple Data Bytes and Read Operation with Multiple Data Bytes (Sr)



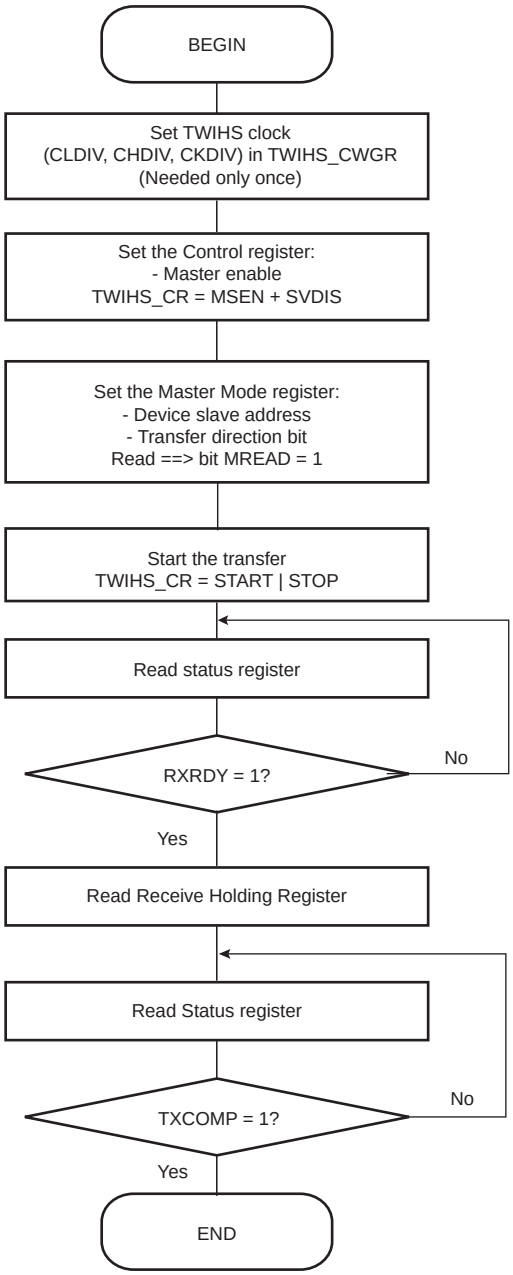
Downloaded from Arrow.com.

Figure 42-20. TWIHS Write Operation with Multiple Data Bytes + Read Operation and Alternative Command Mode + PEC



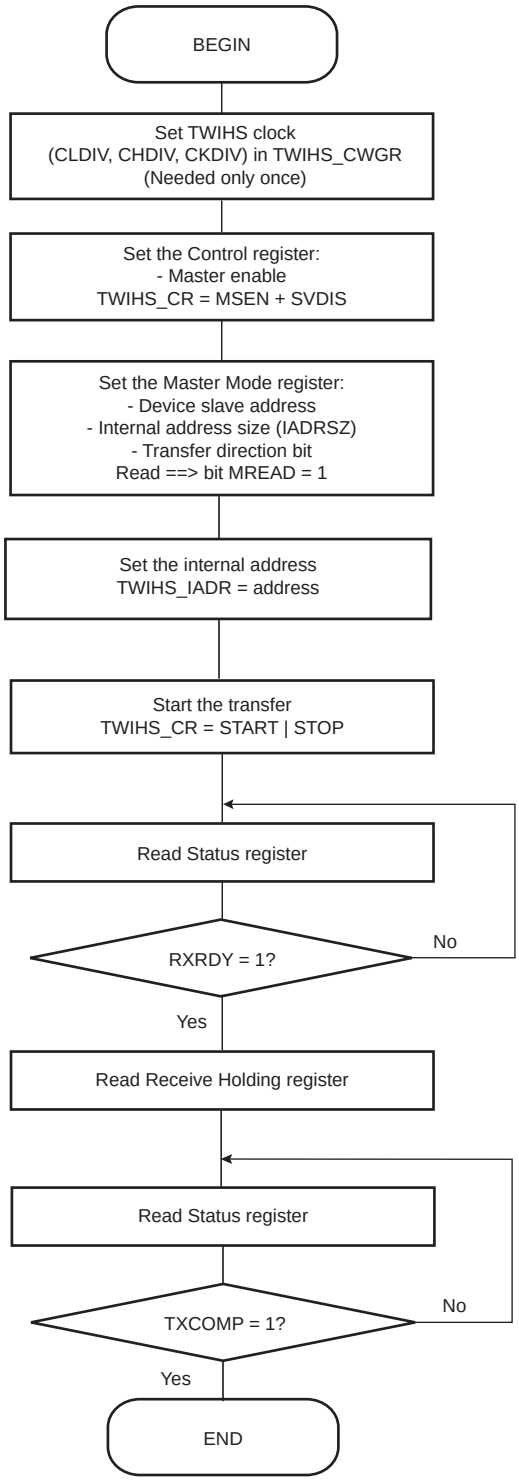
Downloaded from Arrow.com.

Figure 42-21. TWIHS Read Operation with Single Data Byte without Internal Address



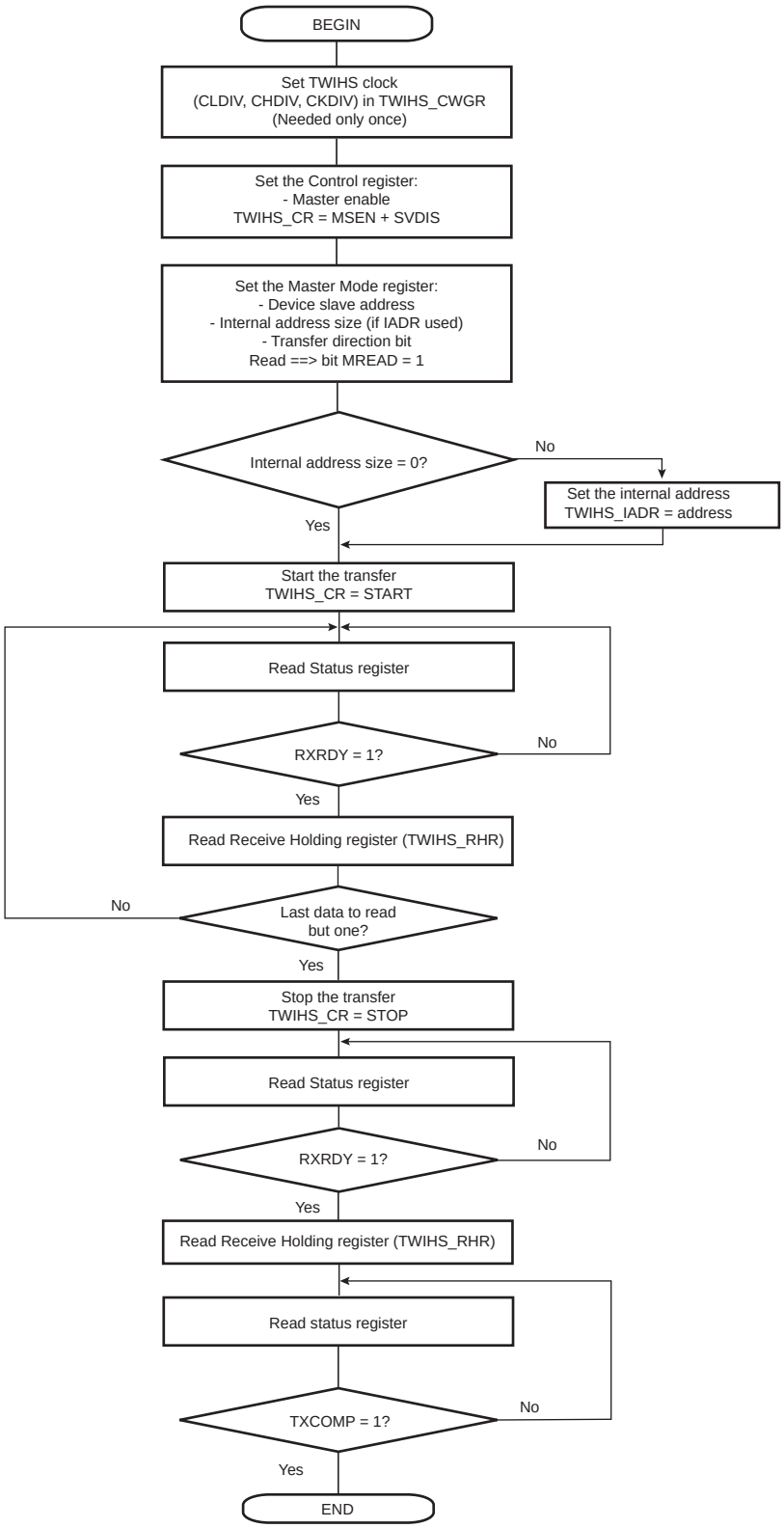
Downloaded from Arrow.com.

Figure 42-22. TWIHS Read Operation with Single Data Byte and Internal Address



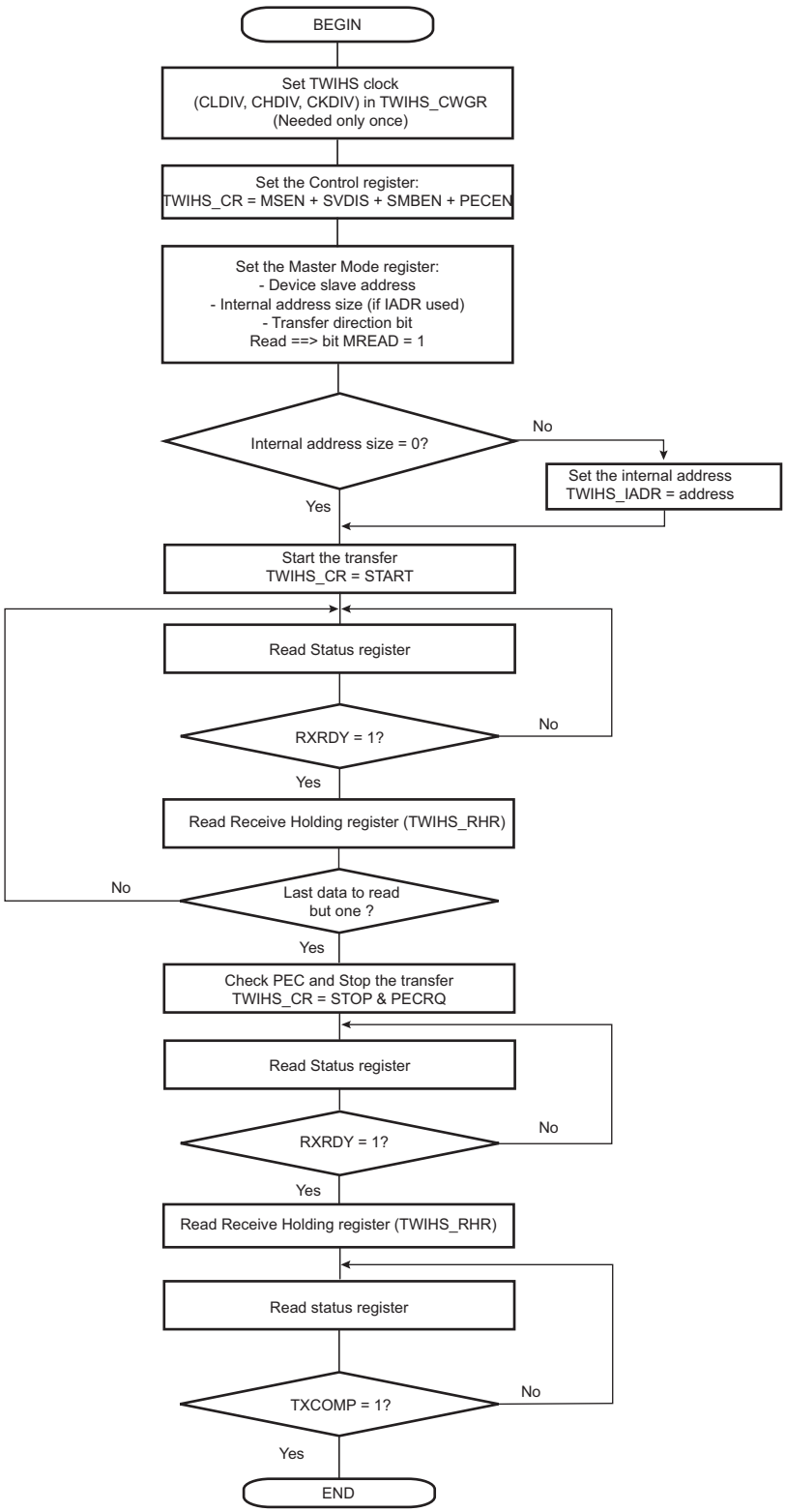
Downloaded from Arrow.com.

Figure 42-23. TWIHS Read Operation with Multiple Data Bytes with or without Internal Address



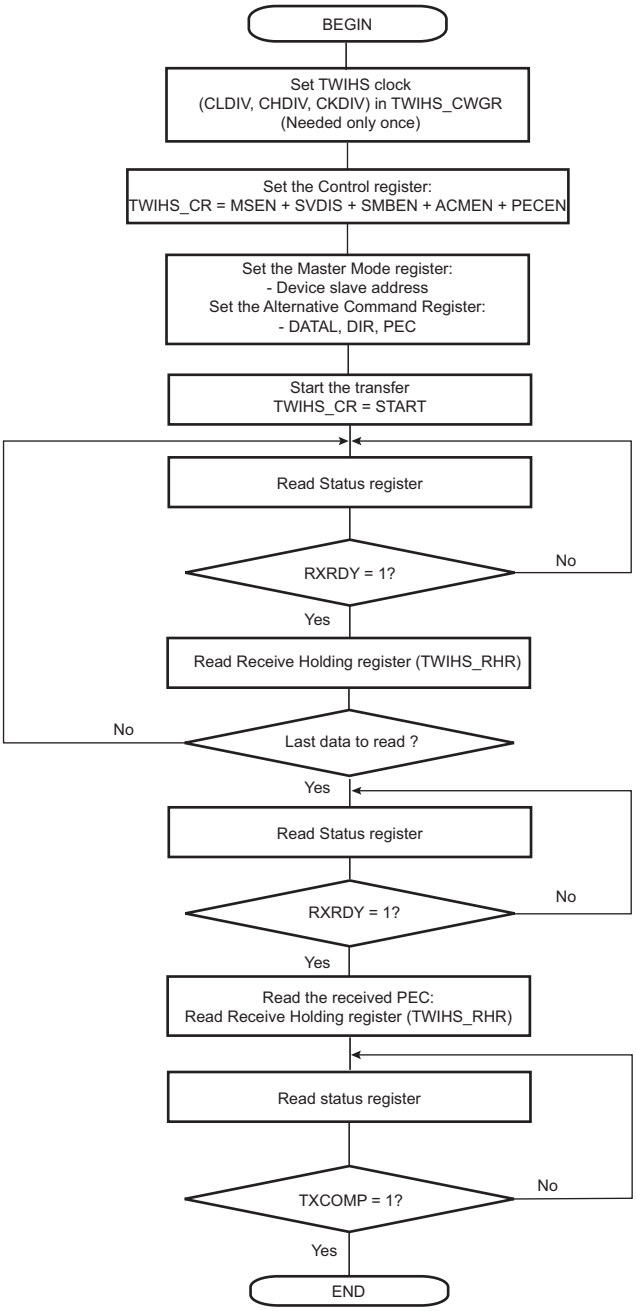
Downloaded from Arrow.com.

Figure 42-24. TWIHS Read Operation with Multiple Data Bytes with or without Internal Address with PEC



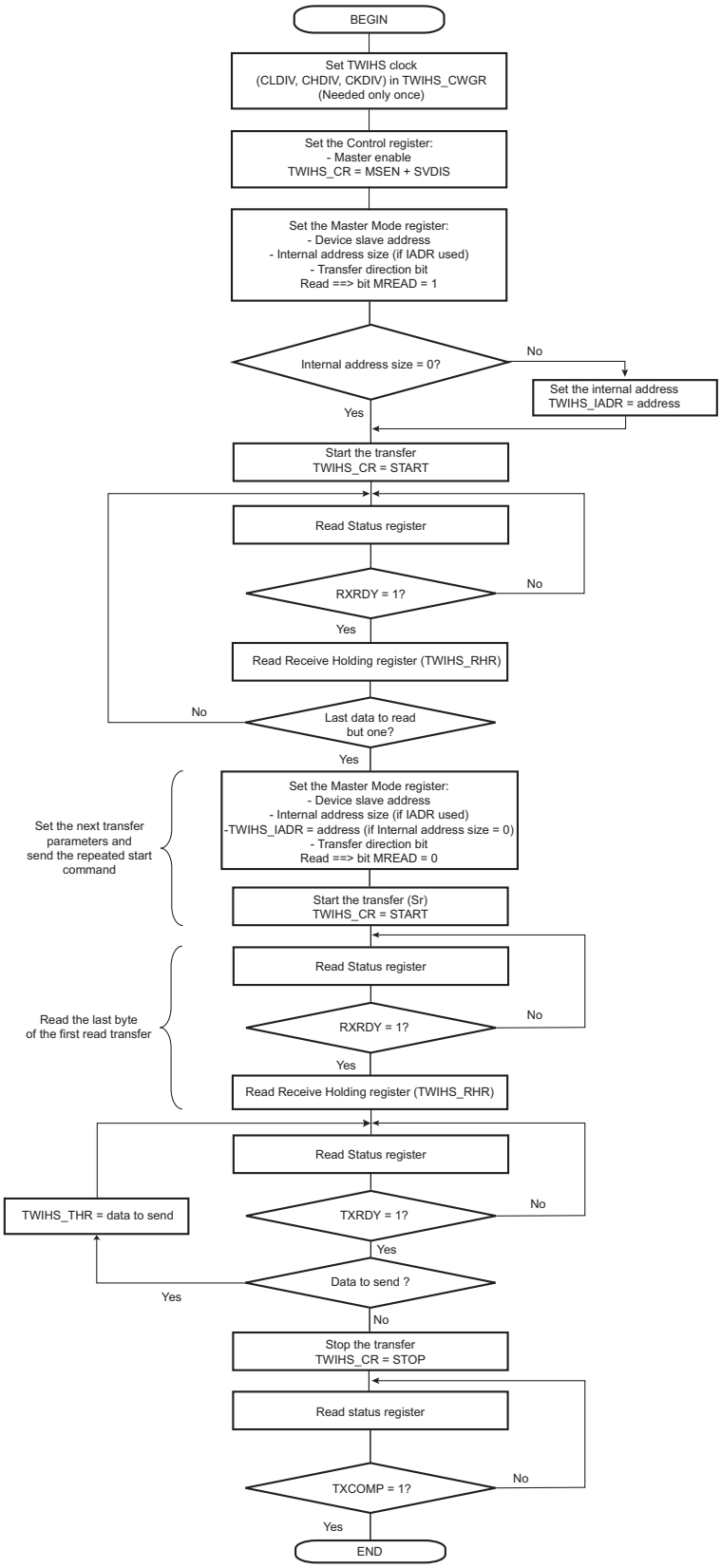
Downloaded from Arrow.com.

Figure 42-25. TWIHS Read Operation with Multiple Data Bytes with Alternative Command Mode with PEC



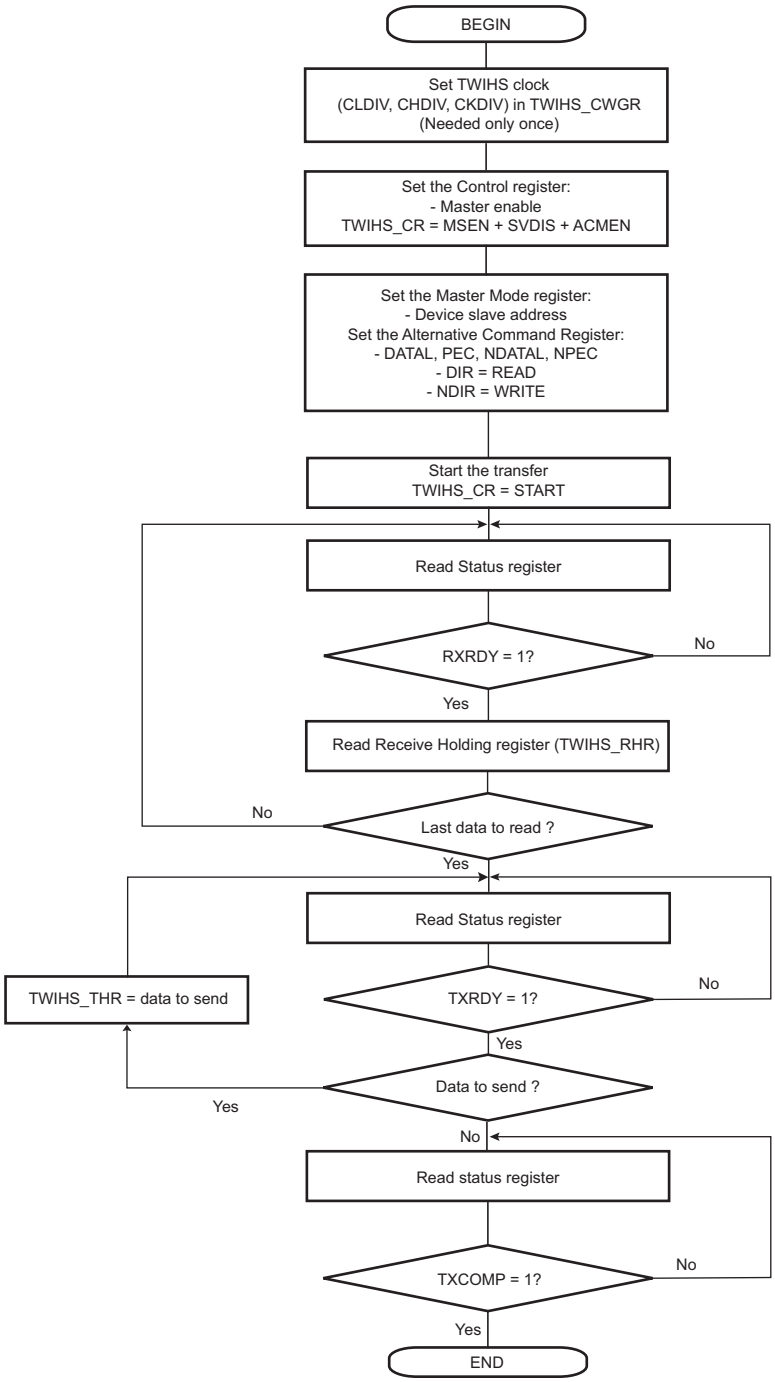
Downloaded from Arrow.com.

Figure 42-26. TWIHS Read Operation with Multiple Data Bytes + Write Operation with Multiple Data Bytes (Sr)



Downloaded from Arrow.com.

Figure 42-27. TWIHS Read Operation with Multiple Data Bytes + Write with Alternative Command Mode with PEC



Downloaded from Arrow.com.

42.6.4 Multi-master Mode

42.6.4.1 Definition

In Multi-master mode, more than one master may handle the bus at the same time without data corruption by using arbitration.

Arbitration starts as soon as two or more masters place information on the bus at the same time, and stops (arbitration is lost) for the master that intends to send a logical one while the other master sends a logical zero.

As soon as arbitration is lost by a master, it stops sending data and listens to the bus in order to detect a stop. When the stop is detected, the master that has lost arbitration may put its data on the bus by respecting arbitration.

Arbitration is illustrated in Figure 42-29.

42.6.4.2 Different Multi-master Modes

Two Multi-master modes may be distinguished:

1. The TWIHS is considered as a master only and is never addressed.
2. The TWIHS may be either a master or a slave and may be addressed.

Note: Arbitration in supported in both Multi-master modes.

TWIHS as Master Only

In this mode, the TWIHS is considered as a master only (MSEN is always at one) and must be driven like a master with the ARBLST (Arbitration Lost) flag in addition.

If arbitration is lost (ARBLST = 1), the user must reinitiate the data transfer.

If starting a transfer (ex.: DADR + START + W + Write in THR) and if the bus is busy, the TWIHS automatically waits for a STOP condition on the bus to initiate the transfer (see Figure 42-28).

Note: The state of the bus (busy or free) is not indicated in the user interface.

TWIHS as Master or Slave

The automatic reversal from master to slave is not supported in case of a lost arbitration.

Then, in the case where TWIHS may be either a master or a slave, the user must manage the pseudo Multi-master mode described in the steps below:

1. Program the TWIHS in Slave mode (SADR + MSDIS + SVEN) and perform a slave access (if TWIHS is addressed).
2. If the TWIHS has to be set in Master mode, wait until TXCOMP flag is at 1.
3. Program the Master mode (DADR + SVDIS + MSEN) and start the transfer (ex: START + Write in THR).
4. As soon as the Master mode is enabled, the TWIHS scans the bus in order to detect if it is busy or free. When the bus is considered free, the TWIHS initiates the transfer.
5. As soon as the transfer is initiated and until a STOP condition is sent, the arbitration becomes relevant and the user must monitor the ARBLST flag.
6. If the arbitration is lost (ARBLST is set to 1), the user must program the TWIHS in Slave mode in case the master that won the arbitration needs to access the TWIHS.
7. If the TWIHS has to be set in Slave mode, wait until the TXCOMP flag is at 1 and then program the Slave mode.

Note: If the arbitration is lost and the TWIHS is addressed, the TWIHS does not acknowledge, even if it is programmed in Slave mode as soon as ARBLST is set to 1. Then the master must repeat SADR.

Downloaded from Arrow.com.

Figure 42-28. User Sends Data While the Bus is Busy

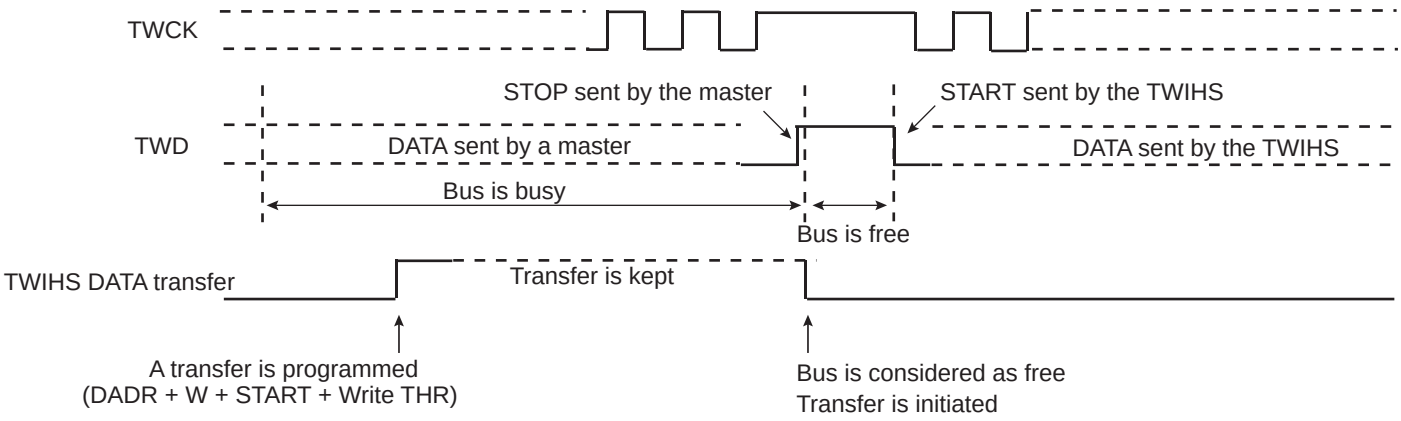
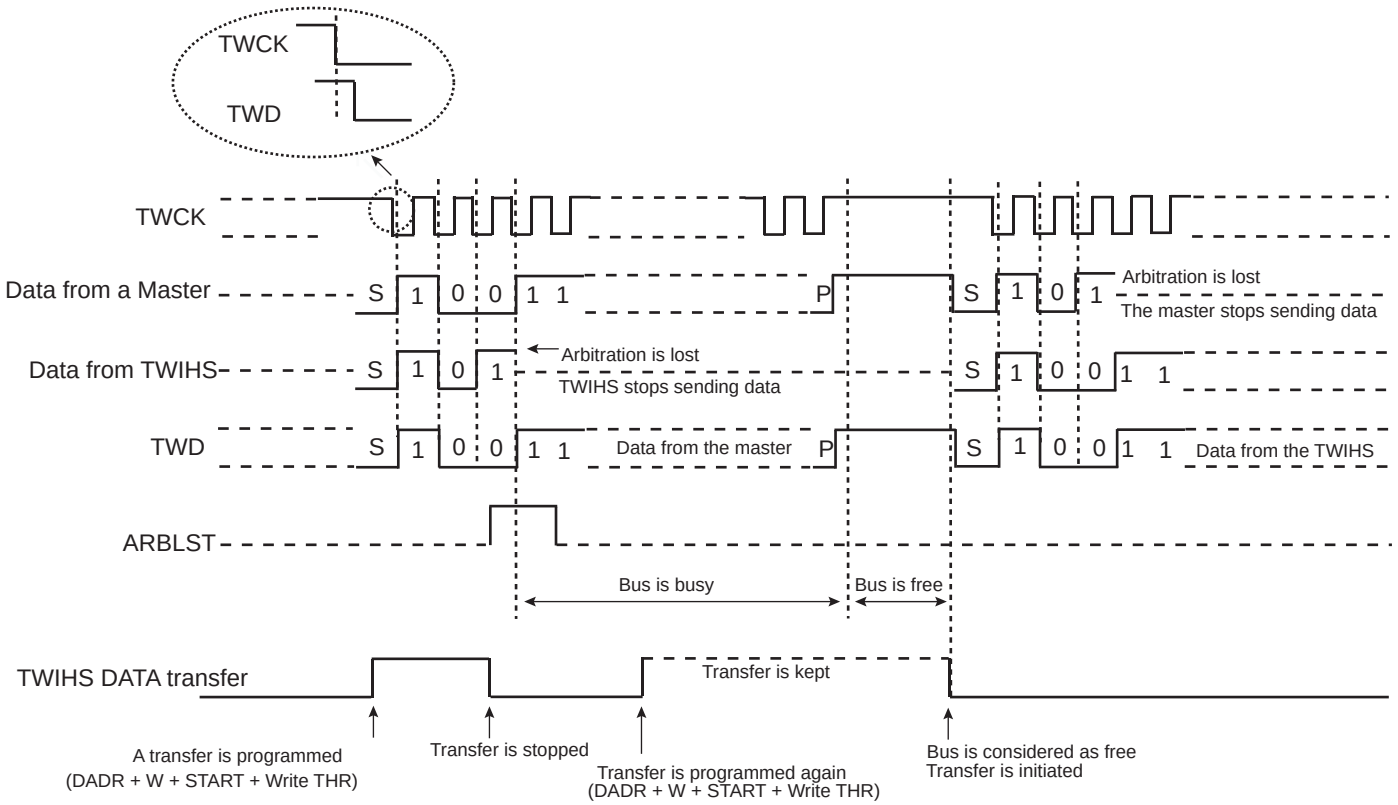


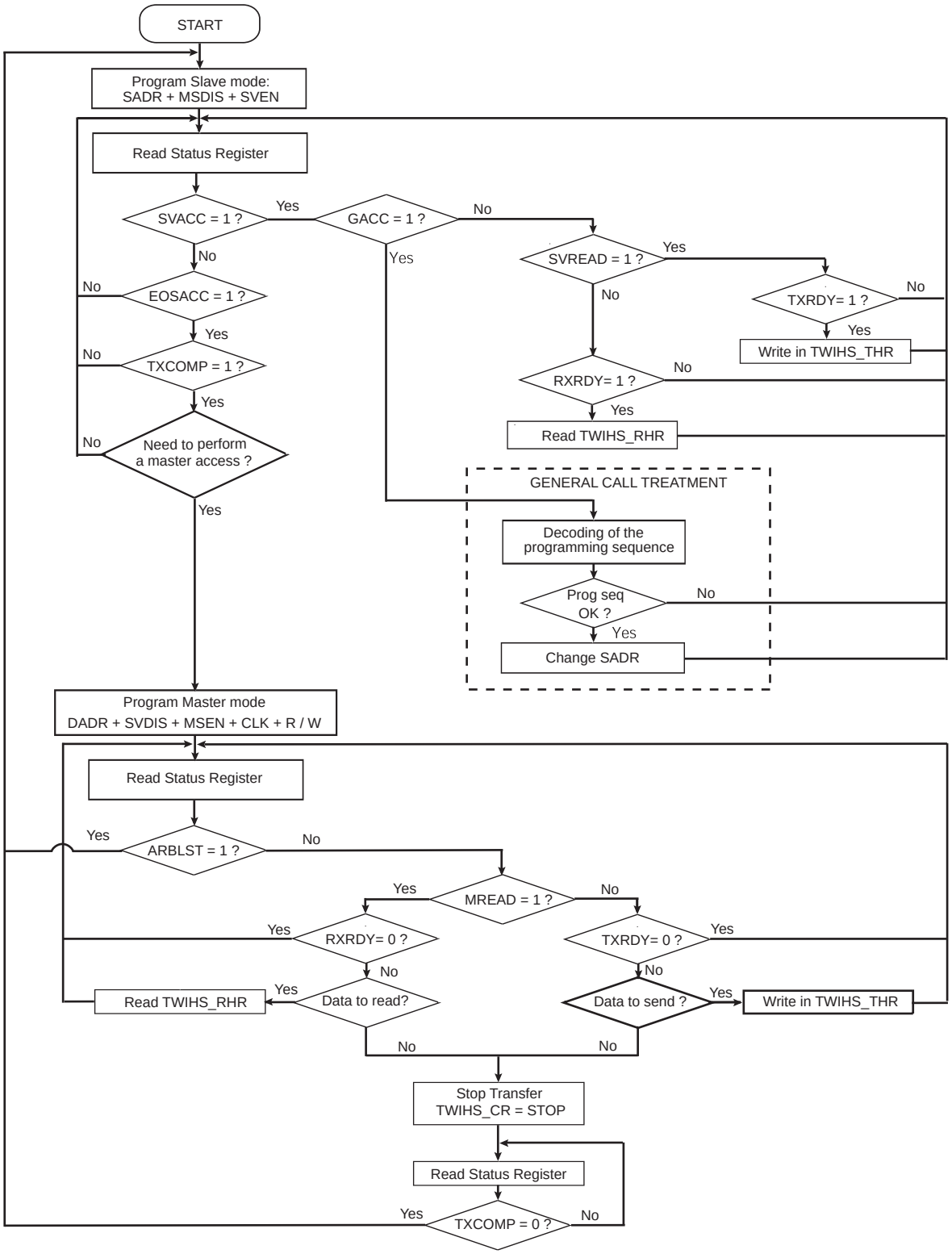
Figure 42-29. Arbitration Cases



The flowchart shown in Figure 42-30 gives an example of read and write operations in Multi-master mode.

Downloaded from Arrow.com.

Figure 42-30. Multi-master Flowchart



Downloaded from Arrow.com.

42.6.5 Slave Mode

42.6.5.1 Definition

Slave mode is defined as a mode where the device receives the clock and the address from another device called the master.

In this mode, the device never initiates and never completes the transmission (START, REPEATED_START and STOP conditions are always provided by the master).

42.6.5.2 Programming Slave Mode

The following fields must be programmed before entering Slave mode:

1. TWIHS_SMR.SADR: The slave device address is used in order to be accessed by master devices in Read or Write mode.
2. (Optional) TWIHS_SMR.MASK can be set to mask some SADR address bits and thus allow multiple address matching.
3. TWIHS_CR.MSDIS: Disables the Master mode.
4. TWIHS_CR.SVEN: Enables the Slave mode.

As the device receives the clock, values written in TWIHS_CWGR are ignored.

42.6.5.3 Receiving Data

After a START or REPEATED START condition is detected, and if the address sent by the master matches the slave address programmed in the SADR (Slave Address) field, the SVACC (Slave Access) flag is set and SVREAD (Slave Read) indicates the direction of the transfer.

SVACC remains high until a STOP condition or a REPEATED START is detected. When such a condition is detected, the EOSACC (End Of Slave Access) flag is set.

Read Sequence

In the case of a read sequence (SVREAD is high), the TWIHS transfers data written in the TWIHS_THR until a STOP condition or a REPEATED_START + an address different from SADR is detected. Note that at the end of the read sequence TXCOMP (Transmission Complete) flag is set and SVACC reset.

As soon as data is written in the TWIHS_THR, TXRDY (Transmit Holding Register Ready) flag is reset, and it is set when the internal shifter is empty and the sent data acknowledged or not. If the data is not acknowledged, the NACK flag is set.

Note that a STOP or a REPEATED START always follows a NACK.

To clear the TXRDY flag, first set the bit TWIHS_CR.SVDIS, then set the bit TWIHS_CR.SVEN.

See Figure 42-31.

Write Sequence

In the case of a write sequence (SVREAD is low), the RXRDY (Receive Holding Register Ready) flag is set as soon as a character has been received in the TWIHS_RHR. RXRDY is reset when reading the TWIHS_RHR.

The TWIHS continues receiving data until a STOP condition or a REPEATED_START + an address different from SADR is detected. Note that at the end of the write sequence, the TXCOMP flag is set and SVACC is reset.

See Figure 42-32.

Downloaded from Arrow.com.

Clock Stretching Sequence

If TWIHS_THR or TWIHS_RHR is not written/read in time, the TWIHS performs a clock stretching.

Clock stretching information is given by the SCLWS (Clock Wait State) bit.

See Figure 42-34 and Figure 42-35.

Note: Clock stretching can be disabled by configuring the SCLWSDIS bit in the TWIHS_SMR. In that case, the UNRE and OVRE flags indicate an underrun (when TWIHS_THR is not filled on time) or an overrun (when TWIHS_RHR is not read on time).

General Call

In the case where a GENERAL CALL is performed, the GACC (General Call Access) flag is set.

After GACC is set, the user must interpret the meaning of the GENERAL CALL and decode the new address programming sequence.

See Figure 42-33.

42.6.5.4 Data Transfer

Read Operation

The Read mode is defined as a data requirement from the master.

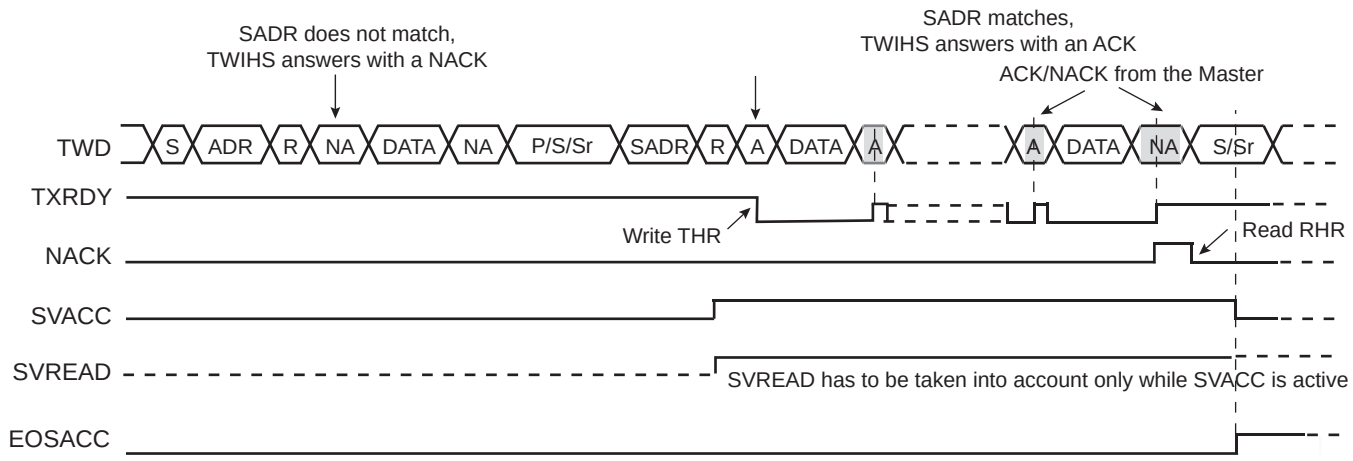
After a START or a REPEATED START condition is detected, the decoding of the address starts. If the slave address (SADR) is decoded, SVACC is set and SVREAD indicates the direction of the transfer.

Until a STOP or REPEATED START condition is detected, the TWIHS continues sending data loaded in the TWIHS_THR.

If a STOP condition or a REPEATED START + an address different from SADR is detected, SVACC is reset.

Figure 42-31 describes the read operation.

Figure 42-31. Read Access Ordered by a Master



Notes:
1. When SVACC is low, the state of SVREAD becomes irrelevant.
2. TXRDY is reset when data has been transmitted from TWIHS_THR to the internal shifter and set when this data has been acknowledged or non acknowledged.

Write Operation

The Write mode is defined as a data transmission from the master.

After a START or a REPEATED START, the decoding of the address starts. If the slave address is decoded, SVACC is set and SVREAD indicates the direction of the transfer (SVREAD is low in this case).

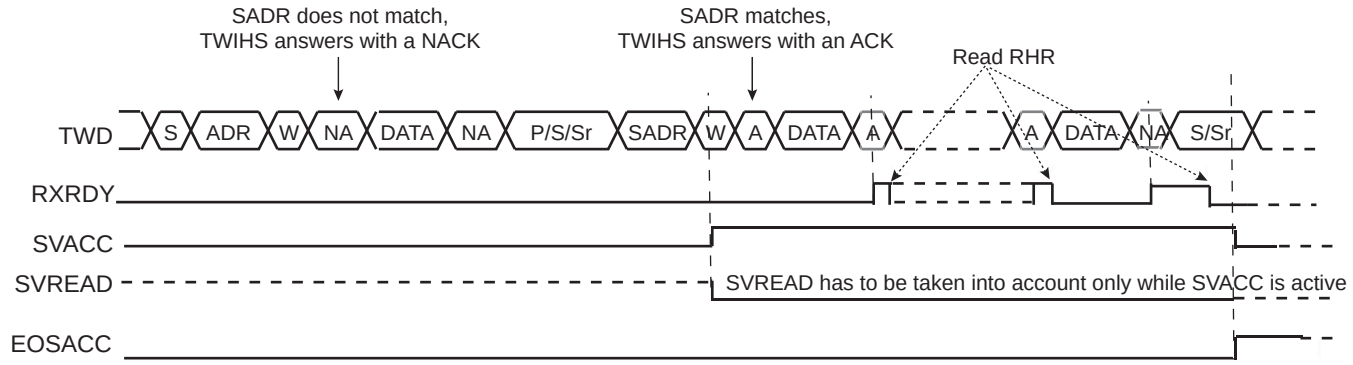
Downloaded from Arrow.com.

Until a STOP or REPEATED START condition is detected, the TWIHS stores the received data in the TWIHS_RHR.

If a STOP condition or a REPEATED START + an address different from SADR is detected, SVACC is reset.

Figure 42-32 describes the write operation.

Figure 42-32. Write Access Ordered by a Master



Notes:
1. When SVACC is low, the state of SVREAD becomes irrelevant.
2. RXRDY is set when data has been transmitted from the internal shifter to the TWIHS_RHR and reset when this data is read.

General Call

The general call is performed in order to change the address of the slave.

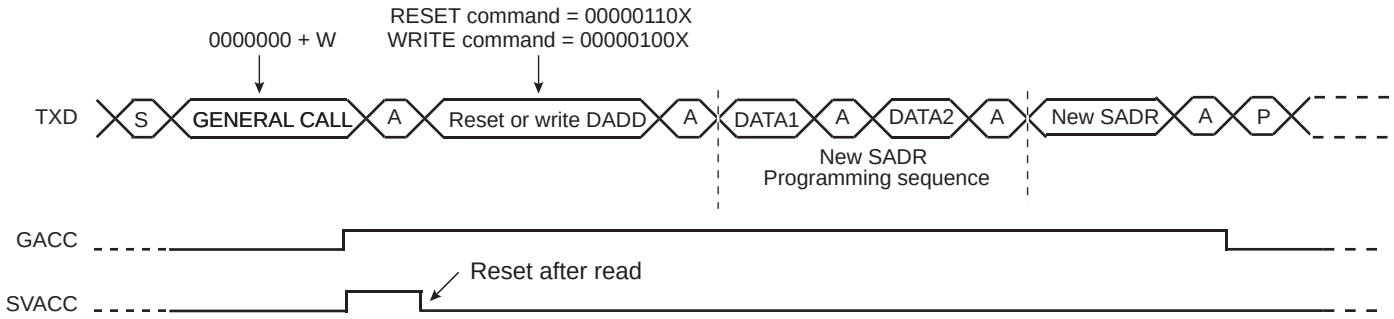
If a GENERAL CALL is detected, GACC is set.

After the detection of general call, decode the commands that follow.

In case of a WRITE command, decode the programming sequence and program a new SADR if the programming sequence matches.

Figure 42-33 describes the general call access.

Figure 42-33. Master Performs a General Call



Note: This method enables the user to create a personal programming sequence by choosing the programming bytes and their number. The programming sequence has to be provided to the master.

Downloaded from Arrow.com.

Clock Stretching

In both Read and Write modes, it may occur that TWIHS_THR/TWIHS_RHR buffer is not filled/emptied before the transmission/reception of a new character. In this case, to avoid sending/receiving undesired data, a clock stretching mechanism is implemented.

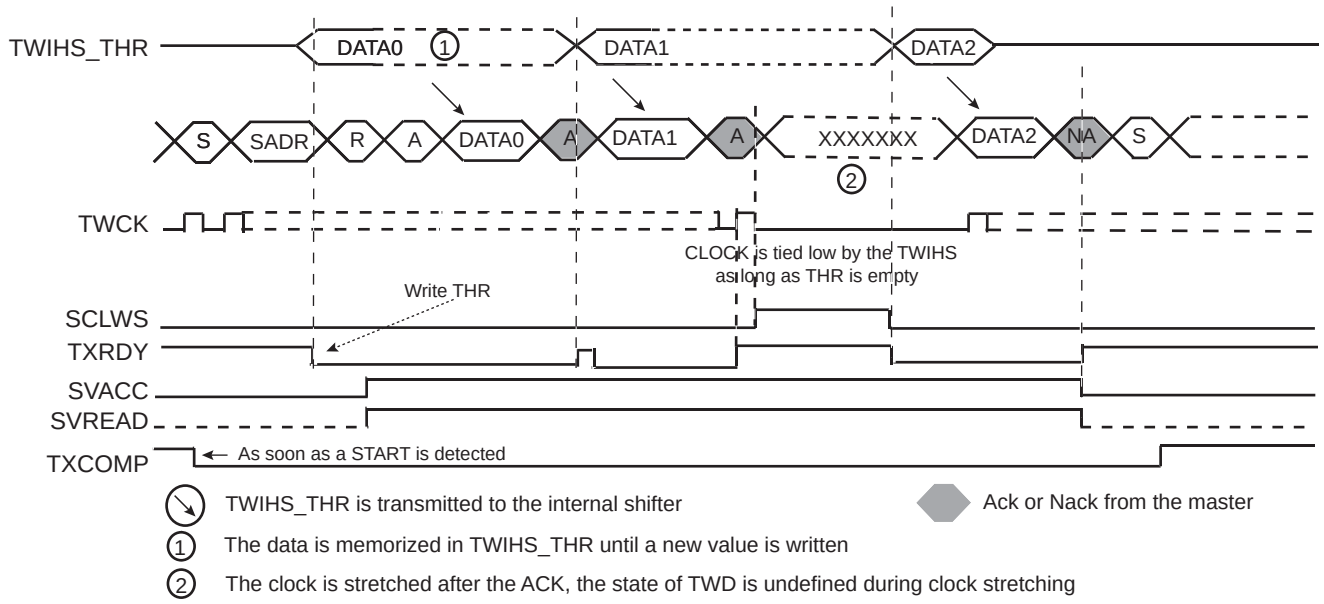
Note: Clock stretching can be disabled by setting the SCLWSDIS bit in the TWIHS_SMR. In that case the UNRE and OVRE flags indicate an underrun (when TWIHS_THR is not filled on time) or an overrun (when TWIHS_RHR is not read on time).

Clock Stretching in Read Mode

The clock is tied low if the internal shifter is empty and if a STOP or REPEATED START condition was not detected. It is tied low until the internal shifter is loaded.

Figure 42-34 describes the clock stretching in Read mode.

Figure 42-34. Clock Stretching in Read Mode



Notes:
1. TXRDY is reset when data has been written in the TWIHS_THR to the internal shifter and set when this data has been acknowledged or non acknowledged.
2. At the end of the read sequence, TXCOMP is set after a STOP or after a REPEATED_START + an address different from SADR.
3. SCLWS is automatically set when the clock stretching mechanism is started.

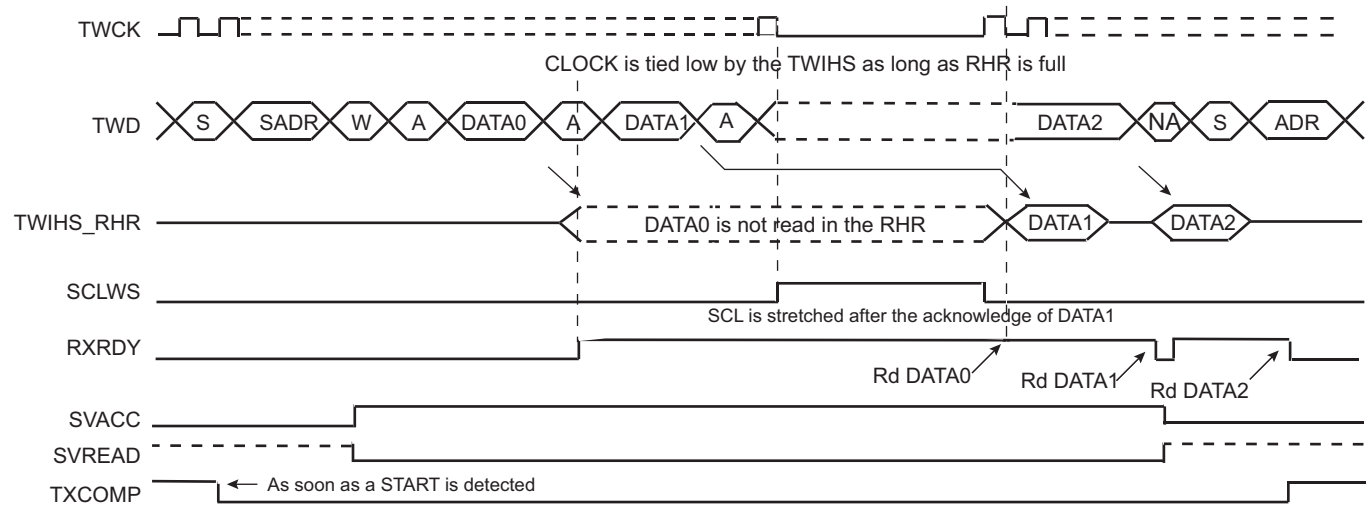
Downloaded from Arrow.com.

Clock Stretching in Write Mode

The clock is tied low if the internal shifter and the TWIHS_RHR is full. If a STOP or REPEATED_START condition was not detected, it is tied low until TWIHS_RHR is read.

Figure 42-35 describes the clock stretching in Write mode.

Figure 42-35. Clock Stretching in Write Mode



Notes:
1. At the end of the read sequence, TXCOMP is set after a STOP or after a REPEATED_START + an address different from SADR.
2. SCLWS is automatically set when the clock stretching mechanism is started and automatically reset when the mechanism is finished.

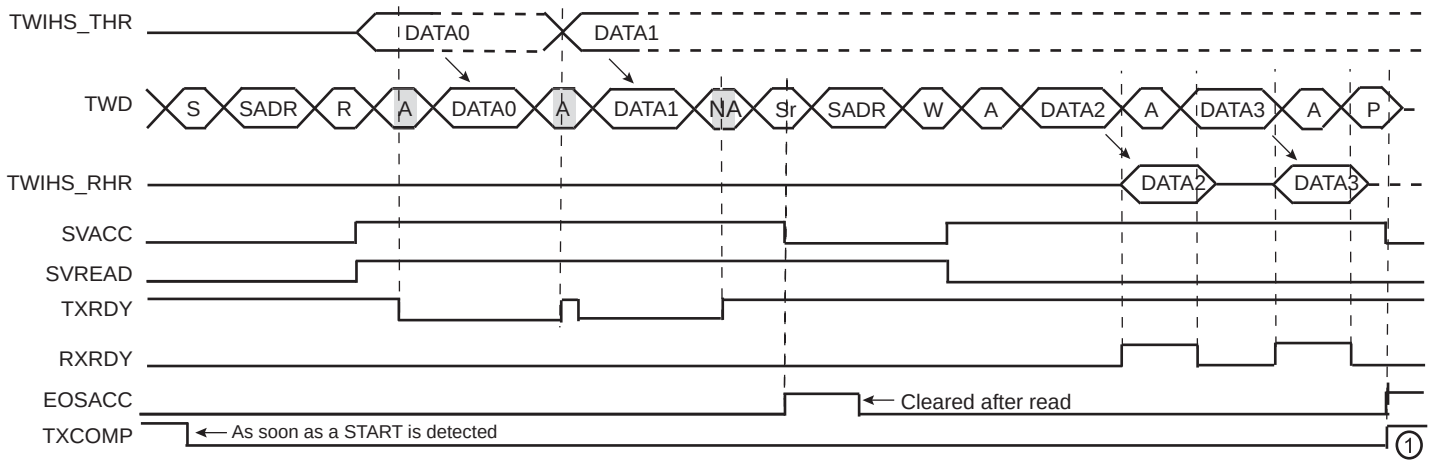
Reversal after a Repeated Start

Reversal of Read to Write

The master initiates the communication by a read command and finishes it by a write command.

Figure 42-36 describes the REPEATED START and the reversal from Read mode to Write mode.

Figure 42-36. Repeated Start and Reversal from Read Mode to Write Mode



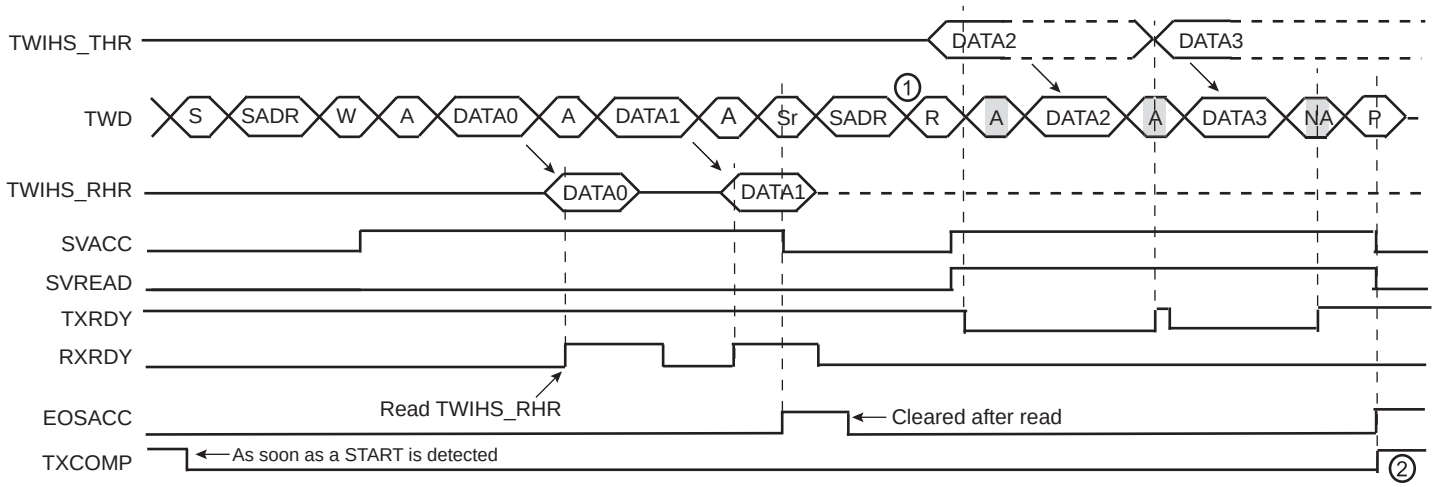
Note: TXCOMP is only set at the end of the transmission. This is because after the REPEATED START, SADR is detected again.

Downloaded from Arrow.com.

Reversal of Write to Read

The master initiates the communication by a write command and finishes it by a read command. Figure 42-37 describes the REPEATED START and the reversal from Write mode to Read mode.

Figure 42-37. Repeated Start and Reversal from Write Mode to Read Mode



Notes:
1. In this case, if TWIHS_THR has not been written at the end of the read command, the clock is automatically stretched before the ACK.
2. TXCOMP is only set at the end of the transmission. This is because after the REPEATED START, SADR is detected again.

42.6.5.5 Using the DMA Controller (DMAC) in Slave Mode

The use of the DMAC significantly reduces the CPU load.

Data Transmit with the DMA in Slave Mode

The following procedure shows an example to transmit data with DMA.

1. Initialize the transmit DMA (memory pointers, transfer size, etc).
2. Configure the Slave mode.
3. Enable the DMA.
4. Wait for the DMA status flag indicating that the buffer transfer is complete.
5. Disable the DMA.
6. (Only if peripheral clock must be disabled) Wait for the TXCOMP flag to be raised in TWIHS_SR.

Data Receive with the DMA in Slave Mode

The following procedure shows an example to transmit data with DMA where the number of characters to receive is known.

1. Initialize the DMA (channels, memory pointers, size, etc.).
2. Configure the Slave mode.
3. Enable the DMA.
4. Wait for the DMA status flag indicating that the buffer transfer is complete.
5. Disable the DMA.
6. (Only if peripheral clock must be disabled) Wait for the TXCOMP flag to be raised in TWIHS_SR.

Downloaded from Arrow.com.

42.6.5.6 SMBus Mode

SMBus mode is enabled when a one is written to the SMEN bit in the TWIHS_CR. SMBus mode operation is similar to I²C operation with the following exceptions:

- Only 7-bit addressing can be used.
- The SMBus standard describes a set of timeout values to ensure progress and throughput on the bus. These timeout values must be programmed into the TWIHS_SMBTR.
- Transmissions can optionally include a CRC byte, called Packet Error Check (PEC).
- A dedicated bus line, SMBALERT, allows a slave to get a master attention.
- A set of addresses have been reserved for protocol handling, such as alert response address (ARA) and host header (HH) address. Address matching on these addresses can be enabled by configuring the TWIHS_CR.

Packet Error Checking

Each SMBus transfer can optionally end with a CRC byte, called the PEC byte. Writing a one to the PECEN bit in TWIHS_CR will send/check the PEC field in the current transfer. The PEC generator is always updated on every bit transmitted or received, so that PEC handling on the following linked transfers is correct.

In Slave Receiver mode, the master calculates a PEC value and transmits it to the slave after all data bytes have been transmitted. Upon reception of this PEC byte, the slave compares it to the PEC value it has computed itself. If the values match, the data was received correctly, and the slave returns an ACK to the master. If the PEC values differ, data was corrupted, and the slave returns a NACK value. The PECERR bit in TWIHS_SR is set automatically if a PEC error occurred.

In Slave Transmitter mode, the slave calculates a PEC value and transmits it to the master after all data bytes have been transmitted. Upon reception of this PEC byte, the master compares it to the PEC value it has computed itself. If the values match, the data was received correctly. If the PEC values differ, data was corrupted, and the master must take appropriate action.

See Section 42.6.5.9 ”Slave Read Write Flowcharts” for detailed flowcharts.

Timeouts

The TWIHS SMBus Timing Register (TWIHS_SMBTR) configures the SMBus timeout values. If a timeout occurs, the slave leaves the bus. The TOUT bit is also set in TWIHS_SR.

42.6.5.7 High-Speed Slave Mode

High-speed mode is enabled when a one is written to the HSEN bit in the TWIHS_CR. Furthermore, the analog pad filter must be enabled, a one must be written to the PADFEN bit in the TWIHS_FILTR and the FILT bit must be cleared. TWIHS High-speed mode operation is similar to TWIHS operation with the following exceptions:

1. A master code is received first at normal speed before entering High-speed mode period.
2. When TWIHS High-speed mode is active, clock stretching is only allowed after acknowledge (ACK), not-acknowledge (NACK), START (S) or REPEATED START (Sr) (as consequence OVF may happen).

TWIHS High-speed mode allows transfers of up to 3.4 Mbit/s.

The TWIHS slave in High-speed mode requires that slave clock stretching is disabled (SCLWSDIS bit at ‘1’). The peripheral clock must run at a minimum of 11 MHz (assuming the system has no latency).

Note: When slave clock stretching is disabled, the TWIHS_RHR must always be read before receiving the next data (MASTER write frame). It is strongly recommended to use either the polling method on the RXRDY flag in TWIHS_SR, or the DMA. If the receive is managed by an interrupt, the TWIHS interrupt priority must be set to the right level and its latency minimized to avoid receive overrun.

Note: When slave clock stretching is disabled, the TWIHS_THR must be filled with the first data to send before the beginning of the frame (MASTER read frame). It is strongly recommended to use either the polling method on the TXRDY flag in TWIHS_SR, or the DMA. If the transmit is managed by an interrupt, the TWIHS interrupt priority must be set to the right level and its latency minimized to avoid transmit underrun.

Downloaded from Arrow.com.

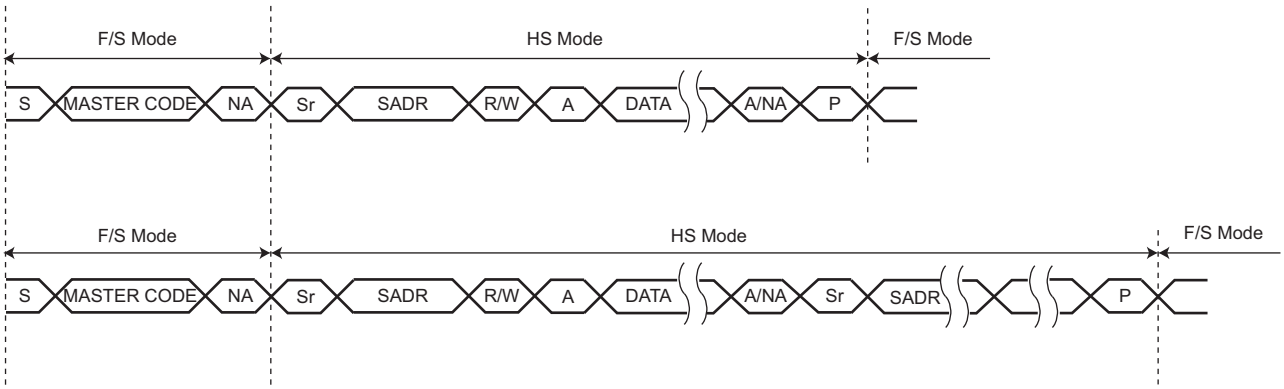
Read/Write Operation

A TWIHS high-speed frame always begins with the following sequence:

1. START condition (S)
2. Master Code (0000 1XXX)
3. Not-acknowledge (NACK)

When the TWIHS is programmed in Slave mode and TWIHS High-speed mode is activated, master code matching is activated and internal timings are set to match the TWIHS High-speed mode requirements.

Figure 42-38. High-Speed Mode Read/Write



Usage

TWIHS High-speed mode usage is the same as the standard TWIHS (See Section 42.6.3.12 "Read/Write Flowcharts").

42.6.5.8 Asynchronous Partial Wake-up (SleepWalking)

The TWIHS includes an asynchronous start condition detector. It is capable of waking the device up from a Sleep mode upon an address match (and optionally an additional data match), including Sleep modes where the TWIHS peripheral clock is stopped.

After detecting the START condition on the bus, the TWIHS stretches TWCK until the TWIHS peripheral clock has started. The time required for starting the TWIHS depends on which Sleep mode the device is in. After the TWIHS peripheral clock has started, the TWIHS releases its TWCK stretching and receives one byte of data (slave address) on the bus. At this time, only a limited part of the device, including the TWIHS module, receives a clock, thus saving power. If the address phase causes a TWIHS address match (and, optionally, if the first data byte causes data match as well), the entire device is waked and normal TWIHS address matching actions are performed. Normal TWIHS transfer then follows. If the TWIHS is not addressed (or if the optional data match fails), the TWIHS peripheral clock is automatically stopped and the device returns to its original Sleep mode.

The TWIHS has the capability to match on more than one address. The SADR1EN, SADR2EN and SADR3EN bits in TWIHS_SMR enable address matching on additional addresses which can be configured through SADR1, SADR2 and SADR3 fields in the TWIHS_SWMR. The SleepWalking matching process can be extended to the first received data byte if DATAMEN bit in TWIHS_SMR is set and, in this case, a complete matching includes address matching and first received data matching. The field DATAM in TWIHS_SWMR configures the data to match on the first received byte.

When the system is in Active mode and the TWIHS enters Asynchronous Partial Wake-up mode, the flag SVACC must be programmed as the unique source of the TWIHS interrupt and the data match comparison must be disabled.

When the system exits Wait mode as the result of a matching condition, the SVACC flag is used to determine if the TWIHS is the source of exit.

Downloaded from Arrow.com.

Figure 42-39. Address Match Only (Data Matching Disabled)

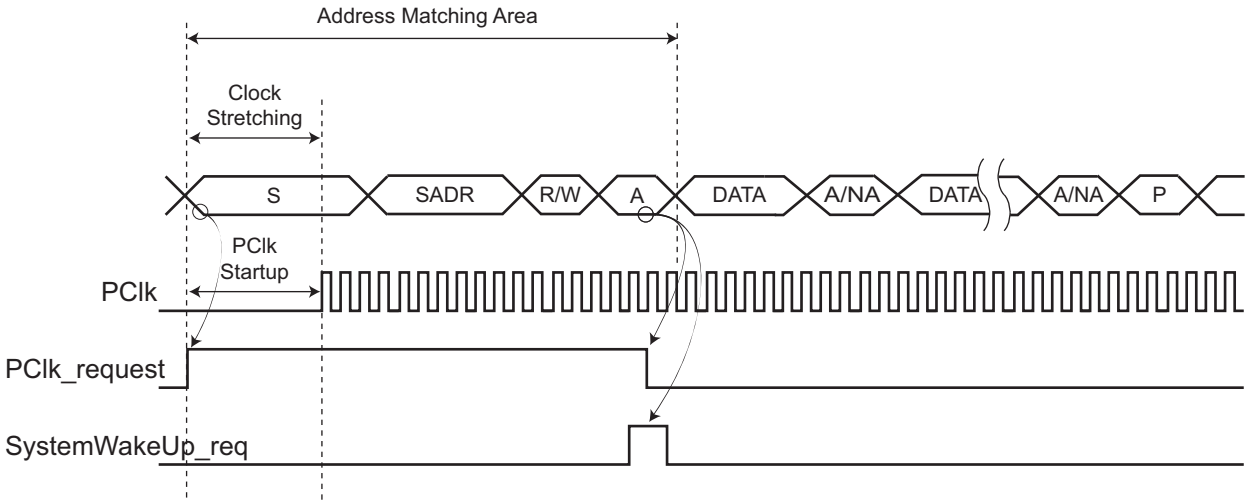
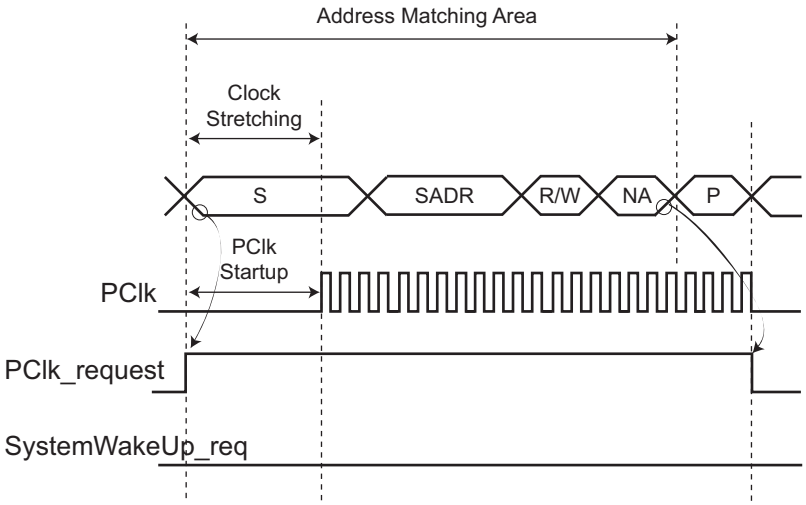


Figure 42-40. No Address Match (Data Matching Disabled)



Downloaded from Arrow.com.

Figure 42-41. Address Match and Data Match (Data Matching Enabled)

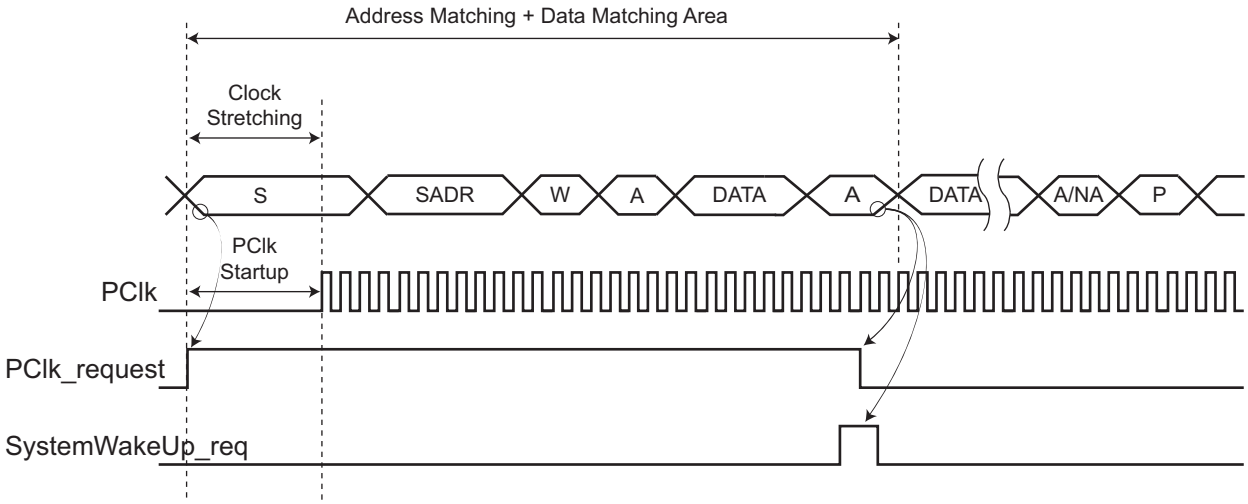
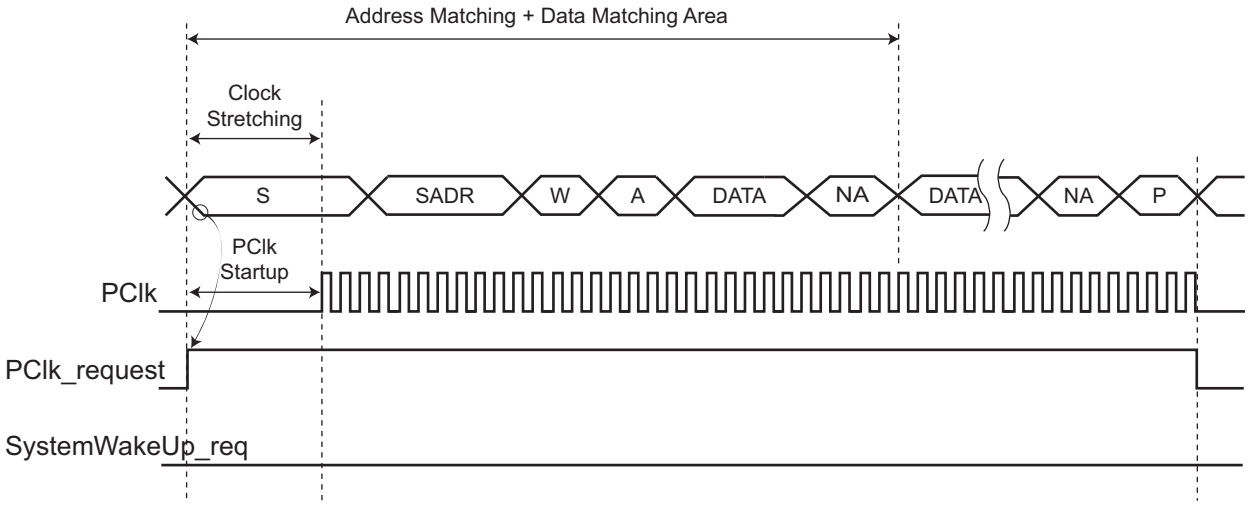


Figure 42-42. Address Match and No Data Match (Data Matching Enabled)

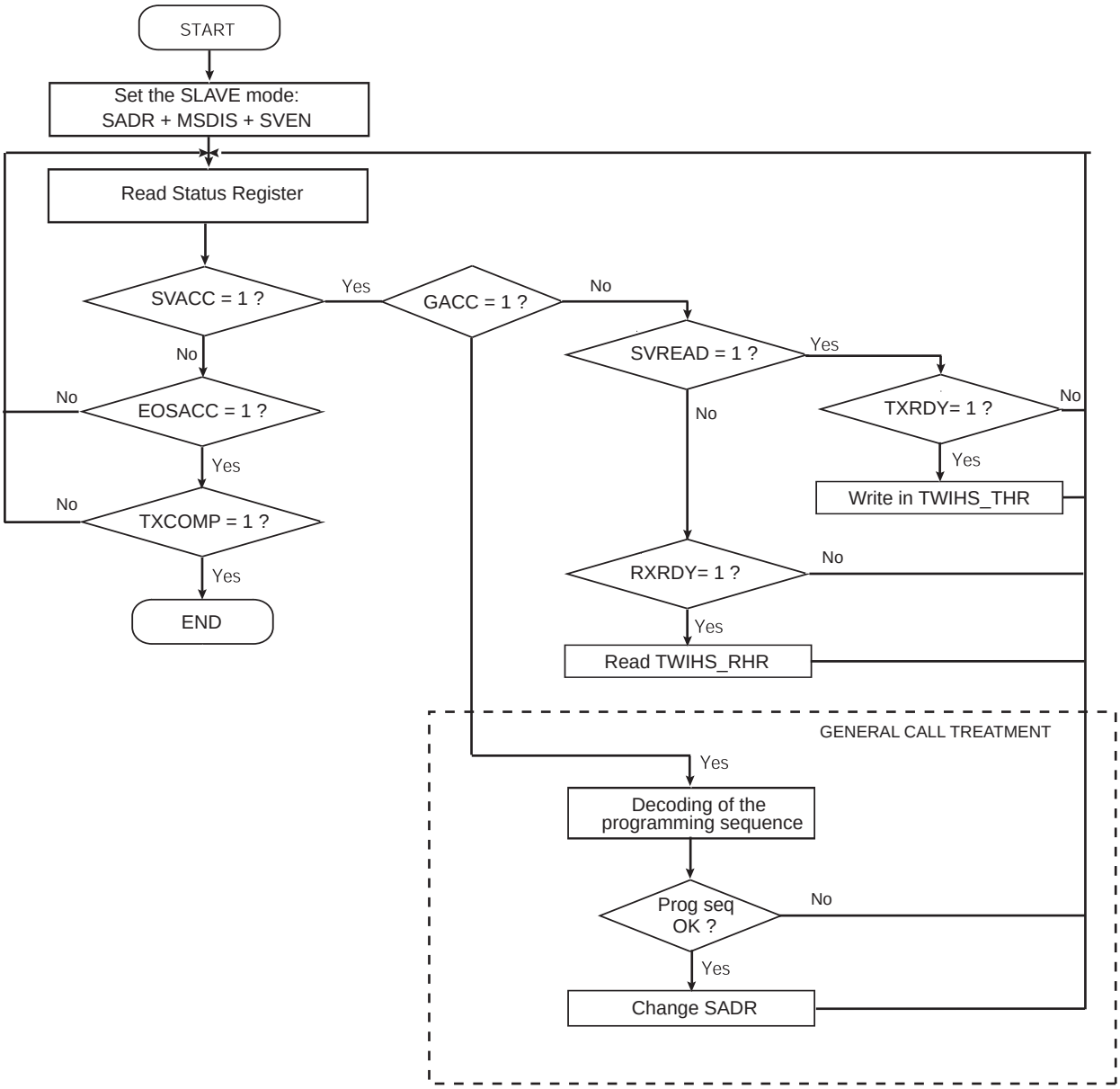


Downloaded from Arrow.com.

42.6.5.9 Slave Read Write Flowcharts

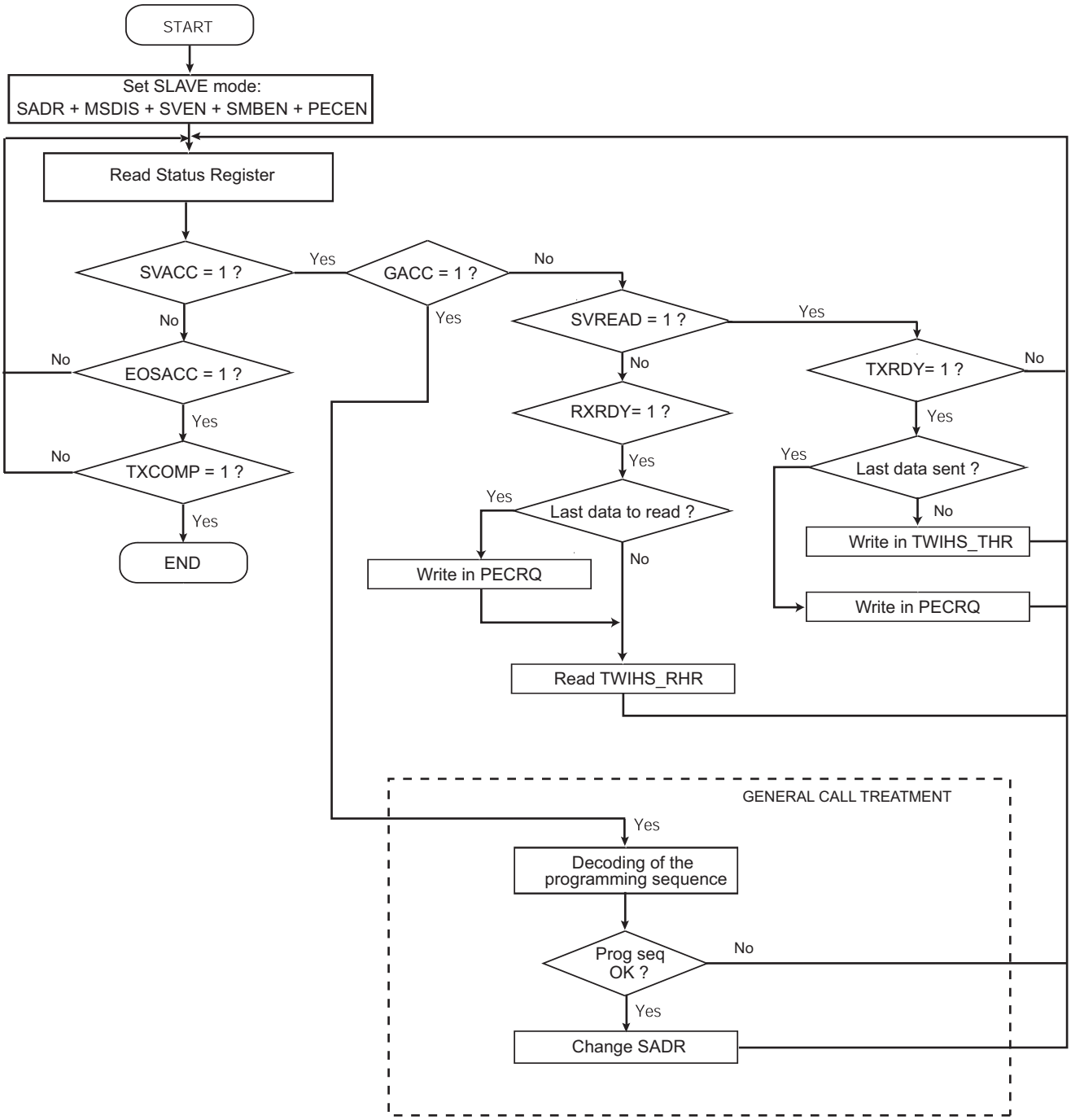
The flowchart shown in Figure 42-43 gives an example of read and write operations in Slave mode. A polling or interrupt method can be used to check the status bits. The interrupt method requires that TWIHS_IER be configured first.

Figure 42-43. Read Write Flowchart in Slave Mode



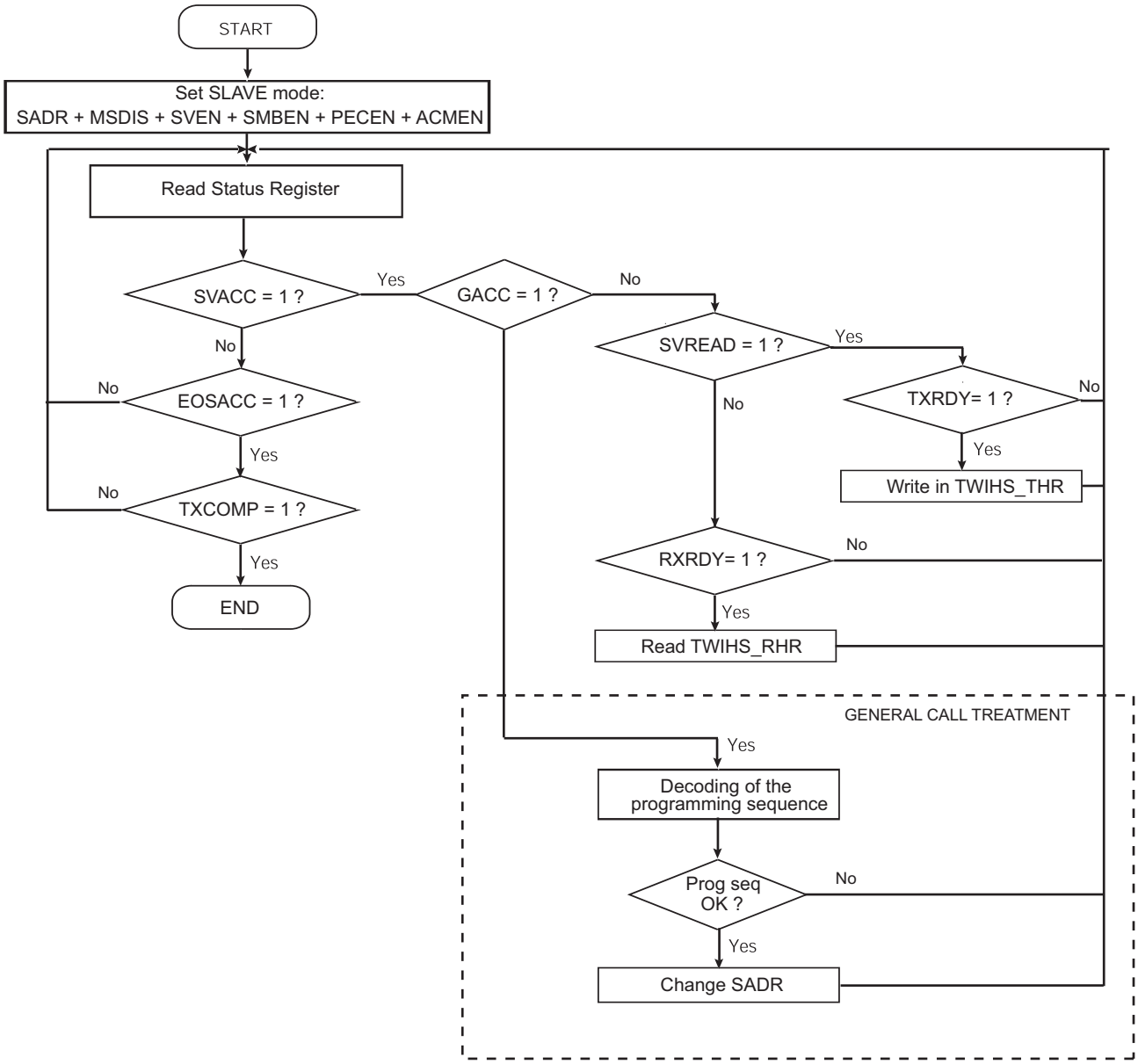
Downloaded from Arrow.com.

Figure 42-44. Read Write Flowchart in Slave Mode with SMBus PEC



Downloaded from Arrow.com.

Figure 42-45. Read Write Flowchart in Slave Mode with SMBus PEC and Alternative Command Mode



42.6.6 TWIHS Comparison Function on Received Character

The comparison function differs if asynchronous partial wake-up (SleepWalking) is enabled or not.

If asynchronous partial wake-up is disabled (see Section 31. "Power Management Controller (PMC)"), the TWIHS can extend the address matching on up to three slave addresses. The SADR1EN, SADR2EN and SADR3EN bits in TWIHS_SMR enable address matching on additional addresses which can be configured through SADR1, SADR2 and SADR3 fields in the TWIHS_SWMR. The DATAMEN bit in the TWIHS_SMR has no effect.

The SVACC bit is set when there is a comparison match with the received slave address.

Downloaded from Arrow.com.

42.6.7 Register Write Protection

To prevent any single software error from corrupting TWIHS behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the TWIHS Write Protection Mode Register (TWIHS_WPMR).

If a write access to a write-protected register is detected, the WPVS bit in the TWIHS Write Protection Status Register (TWIHS_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading TWIHS_WPSR.

The following registers can be write-protected:

- TWIHS Clock Waveform Generator Register

Downloaded from Arrow.com.

42.7 Two-wire Interface High Speed (TWIHS) User Interface

Table 42-7. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	TWIHS_CR	Write-only	–
0x04	Master Mode Register	TWIHS_MMR	Read/Write	0x00000000
0x08	Slave Mode Register	TWIHS_SMR	Read/Write	0x00000000
0x0C	Internal Address Register	TWIHS_IADR	Read/Write	0x00000000
0x10	Clock Waveform Generator Register	TWIHS_CWGR	Read/Write	0x00000000
0x14–0x1C	Reserved	–	–	–
0x20	Status Register	TWIHS_SR	Read-only	0x0300F009
0x24	Interrupt Enable Register	TWIHS_IER	Write-only	–
0x28	Interrupt Disable Register	TWIHS_IDR	Write-only	–
0x2C	Interrupt Mask Register	TWIHS_IMR	Read-only	0x00000000
0x30	Receive Holding Register	TWIHS_RHR	Read-only	0x00000000
0x34	Transmit Holding Register	TWIHS_THR	Write-only	0x00000000
0x38	SMBus Timing Register	TWIHS_SMBTR	Read/Write	0x00000000
0x3C	Reserved	–	–	–
0x40	Reserved	–	–	–
0x44	Filter Register	TWIHS_FILTR	Read/Write	0x00000000
0x48	Reserved	–	–	–
0x4C	SleepWalking Matching Register	TWIHS_SWMR	Read/Write	0x00000000
0x50–0xCC	Reserved	–	–	–
0xD0	Reserved	–	–	–
0xD4–0xE0	Reserved	–	–	–
0xE4	Write Protection Mode Register	TWIHS_WPMR	Read/Write	0x00000000
0xE8	Write Protection Status Register	TWIHS_WPSR	Read-only	0x00000000
0xEC–0xFC[1]	Reserved	–	–	–
0x100–0x128	Reserved	–	–	–

Note: 1. All unlisted offset values are considered as “reserved”.

Downloaded from Arrow.com.

42.7.1 TWIHS Control Register

Name: TWIHS_CR

Address: 0x40018000 (0), 0x4001C000 (1), 0x40060000 (2)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	FIFODIS	FIFOEN	–	LOCKCLR	–	THRCLR
23	22	21	20	19	18	17	16
–	–	–	–	–	–	ACMDIS	ACMEN
15	14	13	12	11	10	9	8
CLEAR	PECRQ	PECDIS	PECEN	SMBDIS	SMBEN	HSDIS	HSEN
7	6	5	4	3	2	1	0
SWRST	QUICK	SVDIS	SVEN	MSDIS	MSEN	STOP	START

- **START: Send a START Condition**

0: No effect.

1: A frame beginning with a START bit is transmitted according to the features defined in the TWIHS Master Mode Register (TWIHS_MMR).

This action is necessary when the TWIHS peripheral needs to read data from a slave. When configured in Master mode with a write operation, a frame is sent as soon as the user writes a character in the Transmit Holding Register (TWIHS_THR).

- **STOP: Send a STOP Condition**

0: No effect.

1: STOP condition is sent just after completing the current byte transmission in Master Read mode.

  – In single data byte master read, both START and STOP must be set.
  – In multiple data bytes master read, the STOP must be set after the last data received but one.
  – In Master Read mode, if a NACK bit is received, the STOP is automatically performed.
  – In master data write operation, a STOP condition will be sent after the transmission of the current data is finished.

- **MSEN: TWIHS Master Mode Enabled**

0: No effect.

1: Enables the Master mode (MSDIS must be written to 0).

Note: Switching from Slave to Master mode is only permitted when TXCOMP = 1.

- **MSDIS: TWIHS Master Mode Disabled**

0: No effect.

1: The Master mode is disabled, all pending data is transmitted. The shifter and holding characters (if it contains data) are transmitted in case of write operation. In read operation, the character being transferred must be completely received before disabling.

Downloaded from Arrow.com.

- **SVEN: TWIHS Slave Mode Enabled**

0: No effect.

1: Enables the Slave mode (SVDIS must be written to 0).

Note: Switching from Master to Slave mode is only permitted when TXCOMP = 1.

- **SVDIS: TWIHS Slave Mode Disabled**

0: No effect.

1: The Slave mode is disabled. The shifter and holding characters (if it contains data) are transmitted in case of read operation. In write operation, the character being transferred must be completely received before disabling.

- **QUICK: SMBus Quick Command**

0: No effect.

1: If Master mode is enabled, a SMBus Quick Command is sent.

- **SWRST: Software Reset**

0: No effect.

1: Equivalent to a system reset.

- **HSEN: TWIHS High-Speed Mode Enabled**

0: No effect.

1: High-speed mode enabled.

- **HSDIS: TWIHS High-Speed Mode Disabled**

0: No effect.

1: High-speed mode disabled.

- **SMBEN: SMBus Mode Enabled**

0: No effect.

1: If SMBDIS = 0, SMBus mode enabled.

- **SMBDIS: SMBus Mode Disabled**

0: No effect.

1: SMBus mode disabled.

- **PECEN: Packet Error Checking Enable**

0: No effect.

1: SMBus PEC (CRC) generation and check enabled.

- **PECDIS: Packet Error Checking Disable**

0: No effect.

1: SMBus PEC (CRC) generation and check disabled.

- **PECRQ: PEC Request**

0: No effect.

1: A PEC check or transmission is requested.

Downloaded from Arrow.com.

- **CLEAR: Bus CLEAR Command**

0: No effect.

1: If Master mode is enabled, sends a bus clear command.

- **ACMEN: Alternative Command Mode Enable**

0: No effect.

1: Alternative Command mode enabled.

- **ACMDIS: Alternative Command Mode Disable**

0: No effect.

1: Alternative Command mode disabled.

- **THRCLR: Transmit Holding Register Clear**

0: No effect.

1: Clears the Transmit Holding Register and set TXRDY, TXCOMP flags.

- **LOCKCLR: Lock Clear**

0: No effect.

1: Clears the TWIHS FSM lock.

- **FIFOEN: FIFO Enable**

0: No effect.

1: Enables the Transmit and Receive FIFOs

- **FIFODIS: FIFO Disable**

0: No effect.

1: Disables the Transmit and Receive FIFOs

Downloaded from Arrow.com.

42.7.2 TWIHS Master Mode Register

Name: TWIHS_MMR

Address: 0x40018004 (0), 0x4001C004 (1), 0x40060004 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	DADR						

15	14	13	12	11	10	9	8
–	–	–	MREAD	–	–	IADRSZ	

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **IADRSZ: Internal Device Address Size**

Value	Name	Description
0	NONE	No internal device address
1	1_BYTE	One-byte internal device address
2	2_BYTE	Two-byte internal device address
3	3_BYTE	Three-byte internal device address

- **MREAD: Master Read Direction**

0: Master write direction.

1: Master read direction.

- **DADR: Device Address**

The device address is used to access slave devices in Read or Write mode. These bits are only used in Master mode.

Downloaded from Arrow.com.

42.7.3 TWIHS Slave Mode Register

Name: TWIHS_SMR

Address: 0x40018008 (0), 0x4001C008 (1), 0x40060008 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
DATAMEN	SADR3EN	SADR2EN	SADR1EN	–	–	–	–
23	22	21	20	19	18	17	16
–	SADR						
15	14	13	12	11	10	9	8
–	MASK						
7	6	5	4	3	2	1	0
–	SCLWSDIS	–	–	SMHH	SMDA	–	NACKEN

- **NACKEN: Slave Receiver Data Phase NACK enable**

0: Normal value to be returned in the ACK cycle of the data phase in Slave Receiver mode.

1: NACK value to be returned in the ACK cycle of the data phase in Slave Receiver mode.

- **SMDA: SMBus Default Address**

0: Acknowledge of the SMBus default address disabled.

1: Acknowledge of the SMBus default address enabled.

- **SMHH: SMBus Host Header**

0: Acknowledge of the SMBus host header disabled.

1: Acknowledge of the SMBus host header enabled.

- **SCLWSDIS: Clock Wait State Disable**

0: No effect.

1: Clock stretching disabled in Slave mode, OVRE and UNRE indicate an overrun/underrun.

- **MASK: Slave Address Mask**

A mask can be applied on the slave device address in Slave mode in order to allow multiple address answer. For each bit of the MASK field set to 1, the corresponding SADR bit is masked.

If the MASK field value is 0, no mask is applied to the SADR field.

- **SADR: Slave Address**

The slave device address is used in Slave mode in order to be accessed by master devices in Read or Write mode.

SADR must be programmed before enabling the Slave mode or after a general call. Writes at other times have no effect.

- **SADR1EN: Slave Address 1 Enable**

0: Slave address 1 matching is disabled.

1: Slave address 1 matching is enabled.

Downloaded from Arrow.com.

- **SADR2EN: Slave Address 2 Enable**

0: Slave address 2 matching is disabled.

1: Slave address 2 matching is enabled.

- **SADR3EN: Slave Address 3 Enable**

0: Slave address 3 matching is disabled.

1: Slave address 3 matching is enabled.

- **DATAMEN: Data Matching Enable**

0: Data matching on first received data is disabled.

1: Data matching on first received data is enabled.

Downloaded from Arrow.com.

42.7.4 TWIHS Internal Address Register

Name: TWIHS_IADR

Address: 0x4001800C (0), 0x4001C00C (1), 0x4006000C (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
IADR							

15	14	13	12	11	10	9	8
IADR							

7	6	5	4	3	2	1	0
IADR							

- **IADR: Internal Address**

0, 1, 2 or 3 bytes depending on IADRSZ.

Downloaded from Arrow.com.

42.7.5 TWIHS Clock Waveform Generator Register

Name: TWIHS_CWGR

Address: 0x40018010 (0), 0x4001C010 (1), 0x40060010 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	HOLD				

23	22	21	20	19	18	17	16
–	–	–	CKSRC	–	CKDIV		

15	14	13	12	11	10	9	8
CHDIV							

7	6	5	4	3	2	1	0
CLDIV							

This register can only be written if the WPEN bit is cleared in the TWIHS Write Protection Mode Register.

TWIHS_CWGR is used in Master mode only.

- **CLDIV: Clock Low Divider**

The SCL low period is defined as follows:

If CKSRC = 0

$t_{low} = ((CLDIV \times 2^{CKDIV}) + 3) \times t_{peripheral\ clock}$

If CKSRC = 1

$t_{low} = (CLDIV \times 2^{CKDIV}) \times t_{external\ clock}$

- **CHDIV: Clock High Divider**

The SCL high period is defined as follows:

If CKSRC = 0

$t_{high} = ((CHDIV \times 2^{CKDIV}) + 3) \times t_{peripheral\ clock}$

If CKSRC = 1

$t_{high} = (CHDIV \times 2^{CKDIV}) \times t_{external\ clock}$

- **CKDIV: Clock Divider**

The CKDIV is used to increase both SCL high and low periods.

- **HOLD: TWD Hold Time Versus TWCK Falling**

If High-speed mode is selected TWD is internally modified on the TWCK falling edge to meet the I2C specified maximum hold time, else if High-speed mode is not configured TWD is kept unchanged after TWCK falling edge for a period of $(HOLD + 3) \times t_{peripheral\ clock}$.

- **CKSRC: Transfer Rate Clock Source**

Value	Name	Description
0	PERIPH_CK	Peripheral clock is used to generate the TWIHS baud rate.
1	PMC_PCK	PMC_PCKx is used to generate the TWIHS baud rate.

Downloaded from Arrow.com.

42.7.6 TWIHS Status Register

Name: TWIHS_SR

Address: 0x40018020 (0), 0x4001C020 (1), 0x40060020 (2)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	SDA	SCL
23	22	21	20	19	18	17	16
–	–	SMBHHM	SMBDAM	PECERR	TOUT	–	MCACK
15	14	13	12	11	10	9	8
–	–	–	–	EOSACC	SCLWS	ARBLST	NACK
7	6	5	4	3	2	1	0
UNRE	OVRE	GACC	SVACC	SVREAD	TXRDY	RXRDY	TXCOMP

Note: This register is not cleared when a debugger performs a read access.

- **TXCOMP: Transmission Completed (cleared by writing TWIHS_THR)**

TXCOMP used in Master mode:

0: During the length of the current frame.

1: When both holding register and internal shifter are empty and STOP condition has been sent.

TXCOMP behavior in Master mode can be seen in Figure 42-6 and in Figure 42-8.

TXCOMP used in Slave mode:

0: As soon as a START is detected.

1: After a STOP or a REPEATED START + an address different from SADR is detected.

TXCOMP behavior in Slave mode can be seen in Figure 42-34, Figure 42-35, Figure 42-36 and Figure 42-37.

- **RXRDY: Receive Holding Register Ready (cleared by reading TWIHS_RHR)**

0: No character has been received since the last TWIHS_RHR read operation.

1: A byte has been received in the TWIHS_RHR since the last read.

RXRDY behavior in Master mode can be seen in Figure 42-7, Figure 42-8 and Figure 42-9.

RXRDY behavior in Slave mode can be seen in Figure 42-32, Figure 42-35, Figure 42-36 and Figure 42-37.

- **TXRDY: Transmit Holding Register Ready (cleared by writing TWIHS_THR)**

TXRDY used in Master mode:

0: The transmit holding register has not been transferred into the internal shifter. Set to 0 when writing into TWIHS_THR.

1: As soon as a data byte is transferred from TWIHS_THR to internal shifter or if a NACK error is detected, TXRDY is set at the same time as TXCOMP and NACK. TXRDY is also set when MSEN is set (enables TWIHS).

TXRDY behavior in Master mode can be seen in Figure 42-4, Figure 42-5 and Figure 42-6.

TXRDY used in Slave mode:

0: As soon as data is written in the TWIHS_THR, until this data has been transmitted and acknowledged (ACK or NACK).

1: Indicates that the TWIHS_THR is empty and that data has been transmitted and acknowledged.

Downloaded from Arrow.com.

If TXRDY is high and if a NACK has been detected, the transmission is stopped. Thus when TRDY = NACK = 1, the user must not fill TWIHS_THR to avoid losing it.

TXRDY behavior in Slave mode can be seen in Figure 42-31, Figure 42-34, Figure 42-36 and Figure 42-37.

- **SVREAD: Slave Read**

This bit is used in Slave mode only. When SVACC is low (no slave access has been detected) SVREAD is irrelevant.

0: Indicates that a write access is performed by a master.

1: Indicates that a read access is performed by a master.

SVREAD behavior can be seen in Figure 42-31, Figure 42-32, Figure 42-36 and Figure 42-37.

- **SVACC: Slave Access**

This bit is used in Slave mode only.

0: TWIHS is not addressed. SVACC is automatically cleared after a NACK or a STOP condition is detected.

1: Indicates that the address decoding sequence has matched (A master has sent SADR). SVACC remains high until a NACK or a STOP condition is detected.

SVACC behavior can be seen in Figure 42-31, Figure 42-32, Figure 42-36 and Figure 42-37.

- **GACC: General Call Access (cleared on read)**

This bit is used in Slave mode only.

0: No general call has been detected.

1: A general call has been detected. After the detection of general call, if need be, the user may acknowledge this access and decode the following bytes and respond according to the value of the bytes.

GACC behavior can be seen in Figure 42-33.

- **OVRE: Overrun Error (cleared on read)**

This bit is used only if clock stretching is disabled.

0: TWIHS_RHR has not been loaded while RXRDY was set.

1: TWIHS_RHR has been loaded while RXRDY was set. Reset by read in TWIHS_SR when TXCOMP is set.

- **UNRE: Underrun Error (cleared on read)**

This bit is used only if clock stretching is disabled.

0: TWIHS_THR has been filled on time.

1: TWIHS_THR has not been filled on time.

- **NACK: Not Acknowledged (cleared on read)**

NACK used in Master mode:

0: Each data byte has been correctly received by the far-end side TWIHS slave component.

1: A data or address byte has not been acknowledged by the slave component. Set at the same time as TXCOMP.

NACK used in Slave Read mode:

0: Each data byte has been correctly received by the master.

1: In Read mode, a data byte has not been acknowledged by the master. When NACK is set, the user must not fill TWIHS_THR even if TXRDY is set, because it means that the master stops the data transfer or re-initiate it.

Note: in Slave Write mode, all data are acknowledged by the TWIHS.

Downloaded from Arrow.com.

- **ARBLST: Arbitration Lost (cleared on read)**

This bit is used in Master mode only.

0: Arbitration won.

1: Arbitration lost. Another master of the TWIHS bus has won the multi-master arbitration. TXCOMP is set at the same time.

- **SCLWS: Clock Wait State**

This bit is used in Slave mode only.

0: The clock is not stretched.

1: The clock is stretched. TWIHS_THR / TWIHS_RHR buffer is not filled / emptied before the transmission / reception of a new character.

SCLWS behavior can be seen in Figure 42-34 and Figure 42-35.

- **EOSACC: End Of Slave Access (cleared on read)**

This bit is used in Slave mode only.

0: A slave access is being performing.

1: The Slave Access is finished. End Of Slave Access is automatically set as soon as SVACC is reset.

EOSACC behavior can be seen in Figure 42-36 and Figure 42-37.

- **MCACK: Master Code Acknowledge (cleared on read)**

MACK used in Slave mode:

0: No Master Code has been received since the last read of TWIHS_SR.

1: A Master Code has been received since the last read of TWIHS_SR.

- **TOUT: Timeout Error (cleared on read)**

0: No SMBus timeout occurred since the last read of TWIHS_SR.

1: An SMBus timeout occurred since the last read of TWIHS_SR.

- **PECERR: PEC Error (cleared on read)**

0: No SMBus PEC error occurred since the last read of TWIHS_SR.

1: An SMBus PEC error occurred since the last read of TWIHS_SR.

- **SMBDAM: SMBus Default Address Match (cleared on read)**

0: No SMBus Default Address received since the last read of TWIHS_SR.

1: An SMBus Default Address was received since the last read of TWIHS_SR.

- **SMBHHM: SMBus Host Header Address Match (cleared on read)**

0: No SMBus Host Header Address received since the last read of TWIHS_SR.

1: An SMBus Host Header Address was received since the last read of TWIHS_SR.

- **SCL: SCL Line Value**

0: SCL line sampled value is ‘0’.

1: SCL line sampled value is ‘1.’

Downloaded from Arrow.com.

- **SDA: SDA Line Value**

0: SDA line sampled value is ‘0’.

1: SDA line sampled value is ‘1’.

Downloaded from Arrow.com.

42.7.7 TWIHS SMBus Timing Register

Name: TWIHS_SMBTR

Address: 0x40018038 (0), 0x4001C038 (1), 0x40060038 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
THMAX							
23	22	21	20	19	18	17	16
TLOWM							
15	14	13	12	11	10	9	8
TLOWS							
7	6	5	4	3	2	1	0
–	–	–	–	PRESC			

- **PRESC: SMBus Clock Prescaler**

Used to specify how to prescale the TLOWS, TLOWM and THMAX counters in SMBTR. Counters are prescaled according to the following formula:

$$f_{Prescaled} = \frac{f_{\text{peripheral clock}}}{2^{(PRESC+1)}}$$

- **TLOWS: Slave Clock Stretch Maximum Cycles**

0: TLOW:SEXT timeout check disabled.

1–255: Clock cycles in slave maximum clock stretch count. Prescaled by PRESC. Used to time TLOW:SEXT.

- **TLOWM: Master Clock Stretch Maximum Cycles**

0: TLOW:MEXT timeout check disabled.

1–255: Clock cycles in master maximum clock stretch count. Prescaled by PRESC. Used to time TLOW:MEXT.

- **THMAX: Clock High Maximum Cycles**

Clock cycles in clock high maximum count. Prescaled by PRESC. Used for bus free detection. Used to time THIGH:MAX.

Downloaded from Arrow.com.

42.7.8 TWIHS Filter Register

Name: TWIHS_FILTR

Address: 0x40018044 (0), 0x4001C044 (1), 0x40060044 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	THRES		
7	6	5	4	3	2	1	0
–	–	–	–	–	PADFCFG	PADFEN	FILT

- **FILT: RX Digital Filter**

0: No filtering applied on TWIHS inputs.

1: TWIHS input filtering is active (only in Standard and Fast modes)

Note: TWIHS digital input filtering follows a majority decision based on three samples from SDA/SCL lines at peripheral clock frequency.

- **PADFEN: PAD Filter Enable**

0: PAD analog filter is disabled.

1: PAD analog filter is enabled. (The analog filter must be enabled if High-speed mode is enabled.)

- **PADFCFG: PAD Filter Config**

See Section 56. ”Electrical Characteristics” for filter configuration details.

- **THRES: Digital Filter Threshold**

0: No filtering applied on TWIHS inputs.

1–7: Maximum pulse width of spikes to be suppressed by the input filter, defined in peripheral clock cycles.

Downloaded from Arrow.com.

42.7.9 TWIHS Interrupt Enable Register

Name: TWIHS_IER

Address: 0x40018024 (0), 0x4001C024 (1), 0x40060024 (2)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	SMBHHM	SMBDAM	PECERR	TOUT	–	MCACK
15	14	13	12	11	10	9	8
–	–	–	–	EOSACC	SCL_WS	ARBLST	NACK
7	6	5	4	3	2	1	0
UNRE	OVRE	GACC	SVACC	–	TXRDY	RXRDY	TXCOMP

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

- **TXCOMP: Transmission Completed Interrupt Enable**
- **RXRDY: Receive Holding Register Ready Interrupt Enable**
- **TXRDY: Transmit Holding Register Ready Interrupt Enable**
- **SVACC: Slave Access Interrupt Enable**
- **GACC: General Call Access Interrupt Enable**
- **OVRE: Overrun Error Interrupt Enable**
- **UNRE: Underrun Error Interrupt Enable**
- **NACK: Not Acknowledge Interrupt Enable**
- **ARBLST: Arbitration Lost Interrupt Enable**
- **SCL_WS: Clock Wait State Interrupt Enable**
- **EOSACC: End Of Slave Access Interrupt Enable**
- **MCACK: Master Code Acknowledge Interrupt Enable**
- **TOUT: Timeout Error Interrupt Enable**
- **PECERR: PEC Error Interrupt Enable**
- **SMBDAM: SMBus Default Address Match Interrupt Enable**
- **SMBHHM: SMBus Host Header Address Match Interrupt Enable**

Downloaded from Arrow.com.

42.7.10 TWIHS Interrupt Disable Register

Name: TWIHS_IDR

Address: 0x40018028 (0), 0x4001C028 (1), 0x40060028 (2)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	SMBHHM	SMBDAM	PECERR	TOUT	–	MCACK
15	14	13	12	11	10	9	8
–	–	–	–	EOSACC	SCL_WS	ARBLST	NACK
7	6	5	4	3	2	1	0
UNRE	OVRE	GACC	SVACC	–	TXRDY	RXRDY	TXCOMP

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

- **TXCOMP: Transmission Completed Interrupt Disable**
- **RXRDY: Receive Holding Register Ready Interrupt Disable**
- **TXRDY: Transmit Holding Register Ready Interrupt Disable**
- **SVACC: Slave Access Interrupt Disable**
- **GACC: General Call Access Interrupt Disable**
- **OVRE: Overrun Error Interrupt Disable**
- **UNRE: Underrun Error Interrupt Disable**
- **NACK: Not Acknowledge Interrupt Disable**
- **ARBLST: Arbitration Lost Interrupt Disable**
- **SCL_WS: Clock Wait State Interrupt Disable**
- **EOSACC: End Of Slave Access Interrupt Disable**
- **MCACK: Master Code Acknowledge Interrupt Disable**
- **TOUT: Timeout Error Interrupt Disable**
- **PECERR: PEC Error Interrupt Disable**
- **SMBDAM: SMBus Default Address Match Interrupt Disable**
- **SMBHHM: SMBus Host Header Address Match Interrupt Disable**

Downloaded from Arrow.com.

42.7.11 TWIHS Interrupt Mask Register

Name: TWIHS_IMR

Address: 0x4001802C (0), 0x4001C02C (1), 0x4006002C (2)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	SMBHHM	SMBDAM	PECERR	TOUT	–	MCACK
15	14	13	12	11	10	9	8
–	–	–	–	EOSACC	SCL_WS	ARBLST	NACK
7	6	5	4	3	2	1	0
UNRE	OVRE	GACC	SVACC	–	TXRDY	RXRDY	TXCOMP

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

- **TXCOMP: Transmission Completed Interrupt Mask**
- **RXRDY: Receive Holding Register Ready Interrupt Mask**
- **TXRDY: Transmit Holding Register Ready Interrupt Mask**
- **SVACC: Slave Access Interrupt Mask**
- **GACC: General Call Access Interrupt Mask**
- **OVRE: Overrun Error Interrupt Mask**
- **UNRE: Underrun Error Interrupt Mask**
- **NACK: Not Acknowledge Interrupt Mask**
- **ARBLST: Arbitration Lost Interrupt Mask**
- **SCL_WS: Clock Wait State Interrupt Mask**
- **EOSACC: End Of Slave Access Interrupt Mask**
- **MCACK: Master Code Acknowledge Interrupt Mask**
- **TOUT: Timeout Error Interrupt Mask**
- **PECERR: PEC Error Interrupt Mask**
- **SMBDAM: SMBus Default Address Match Interrupt Mask**
- **SMBHHM: SMBus Host Header Address Match Interrupt Mask**

Downloaded from Arrow.com.

42.7.12 TWIHS Receive Holding Register

Name: TWIHS_RHR

Address: 0x40018030 (0), 0x4001C030 (1), 0x40060030 (2)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
RXDATA							

Note: This register is not cleared when a debugger performs a read access.

- **RXDATA: Master or Slave Receive Holding Data**

Downloaded from Arrow.com.

42.7.13 TWIHS SleepWalking Matching Register

Name: TWIHS_SWMR

Address: 0x4001804C (0), 0x4001C04C (1), 0x4006004C (2)

Access: Read/Write

31	30	29	28	27	26	25	24
DATAM							

23	22	21	20	19	18	17	16
–	SADR3						

15	14	13	12	11	10	9	8
–	SADR2						

7	6	5	4	3	2	1	0
–	SADR1						

- **SADR1: Slave Address 1**

Slave address 1. The TWIHS module matches on this additional address if SADR1EN bit is enabled.

- **SADR2: Slave Address 2**

Slave address 2. The TWIHS module matches on this additional address if SADR2EN bit is enabled.

- **SADR3: Slave Address 3**

Slave address 3. The TWIHS module matches on this additional address if SADR3EN bit is enabled.

- **DATAM: Data Match**

The TWIHS module extends the SleepWalking matching process to the first received data, comparing it with DATAM if DATAMEN bit is enabled.

Downloaded from Arrow.com.

42.7.14 TWIHS Transmit Holding Register

Name: TWIHS_THR

Address: 0x40018034 (0), 0x4001C034 (1), 0x40060034 (2)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
TXDATA							

- **TXDATA: Master or Slave Transmit Holding Data**

Downloaded from Arrow.com.

42.7.15 TWIHS Write Protection Mode Register

Name: TWIHS_WPMR

Address: 0x400180E4 (0), 0x4001C0E4 (1), 0x400600E4 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							

23	22	21	20	19	18	17	16
WPKEY							

15	14	13	12	11	10	9	8
WPKEY							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x545749 (“TWI” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x545749 (“TWI” in ASCII).

See Section 42.6.7 ”Register Write Protection” for the list of registers that can be write-protected.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x545749	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0

Downloaded from Arrow.com.

42.7.16 TWIHS Write Protection Status Register

Name: TWIHS_WPSR

Address: 0x400180E8 (0), 0x4001C0E8 (1), 0x400600E8 (2)

Access: Read-only

31	30	29	28	27	26	25	24
WPVSRC							

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

Note: This register is not cleared when a debugger performs a read access.

- **WPVS: Write Protection Violation Status**

0: No write protection violation has occurred since the last read of the TWIHS_WPSR.

1: A write protection violation has occurred since the last read of the TWIHS_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protection Violation Source**

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

Downloaded from Arrow.com.

43. Synchronous Serial Controller (SSC)

43.1 Description

The Synchronous Serial Controller (SSC) provides a synchronous communication link with external devices. It supports many serial synchronous communication protocols generally used in audio and telecom applications such as I2S, Short Frame Sync, Long Frame Sync, etc.

The SSC contains an independent receiver and transmitter and a common clock divider. The receiver and the transmitter each interface with three signals: the TD/RD signal for data, the TK/RK signal for the clock and the TF/RF signal for the Frame Sync. The transfers can be programmed to start automatically or on different events detected on the Frame Sync signal.

The SSC high-level of programmability and its use of DMA permit a continuous high bit rate data transfer without processor intervention.

Featuring connection to the DMA, the SSC permits interfacing with low processor overhead to the following:

- Codecs in master or slave mode
- DAC through dedicated serial interface, particularly I2S
- Magnetic card reader

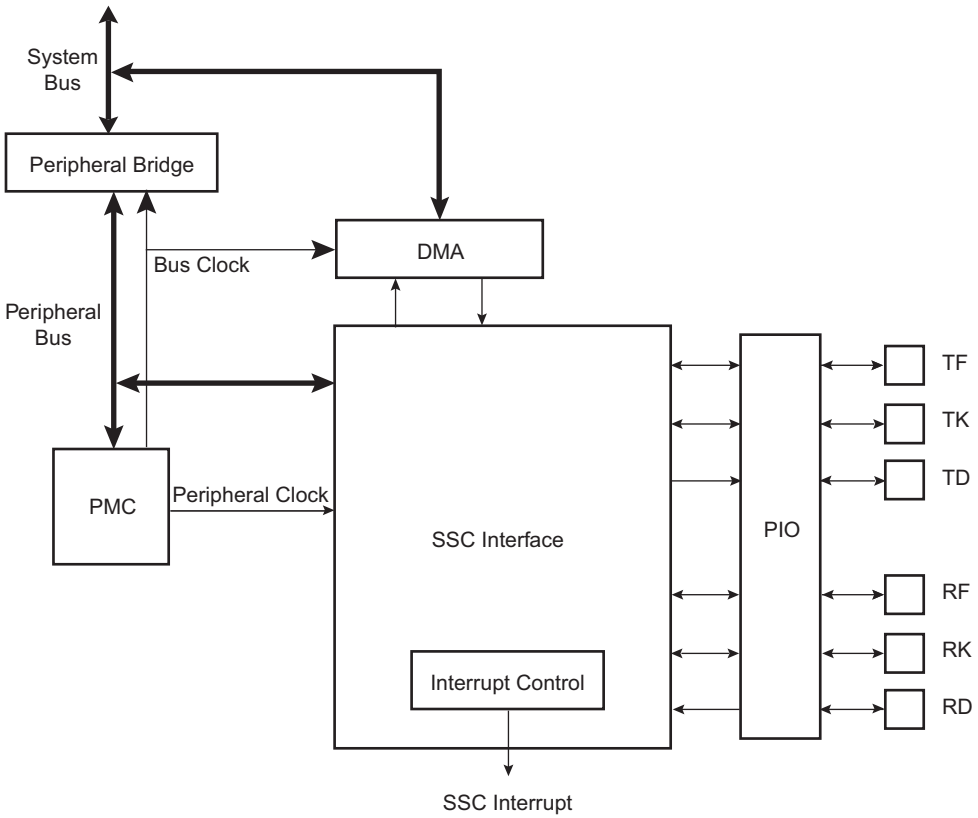
43.2 Embedded Characteristics

- Provides Serial Synchronous Communication Links Used in Audio and Telecom Applications
- Contains an Independent Receiver and Transmitter and a Common Clock Divider
- Interfaced with the DMA Controller (DMAC) to Reduce Processor Overhead
- Offers a Configurable Frame Sync and Data Length
- Receiver and Transmitter Can be Programmed to Start Automatically or on Detection of Different Events on the Frame Sync Signal
- Receiver and Transmitter Include a Data Signal, a Clock Signal and a Frame Synchronization Signal

Downloaded from Arrow.com.

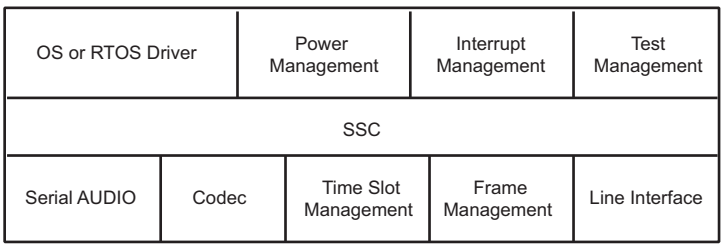
43.3 Block Diagram

Figure 43-1. Block Diagram



43.4 Application Block Diagram

Figure 43-2. Application Block Diagram



Downloaded from Arrow.com.

43.5 SSC Application Examples

The SSC can support several serial communication modes used in audio or high speed serial links. Some standard applications are shown in the following figures. All serial link applications supported by the SSC are not listed here.

Figure 43-3. Audio Application Block Diagram

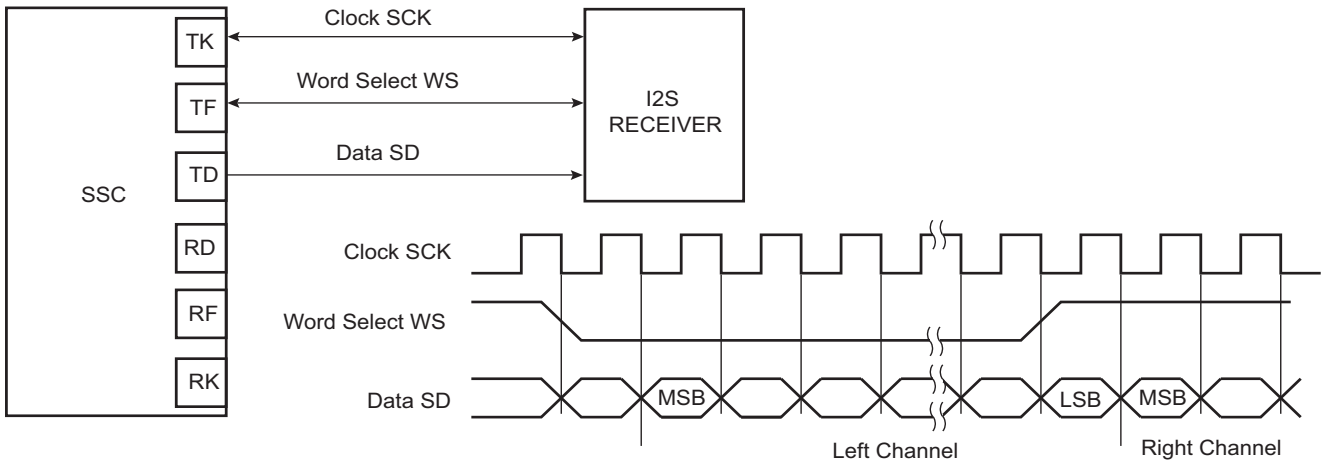
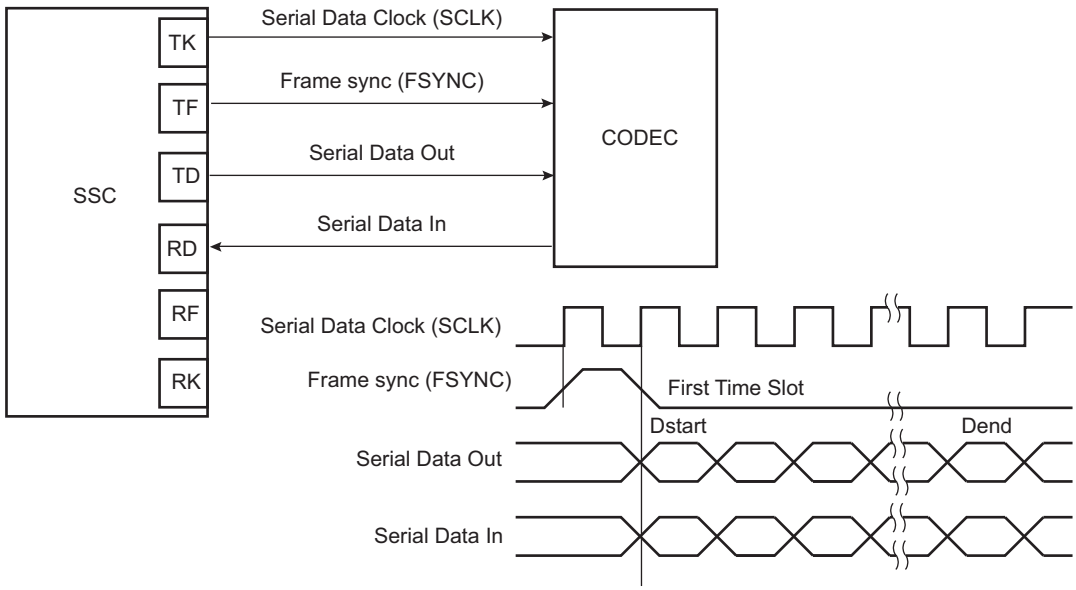
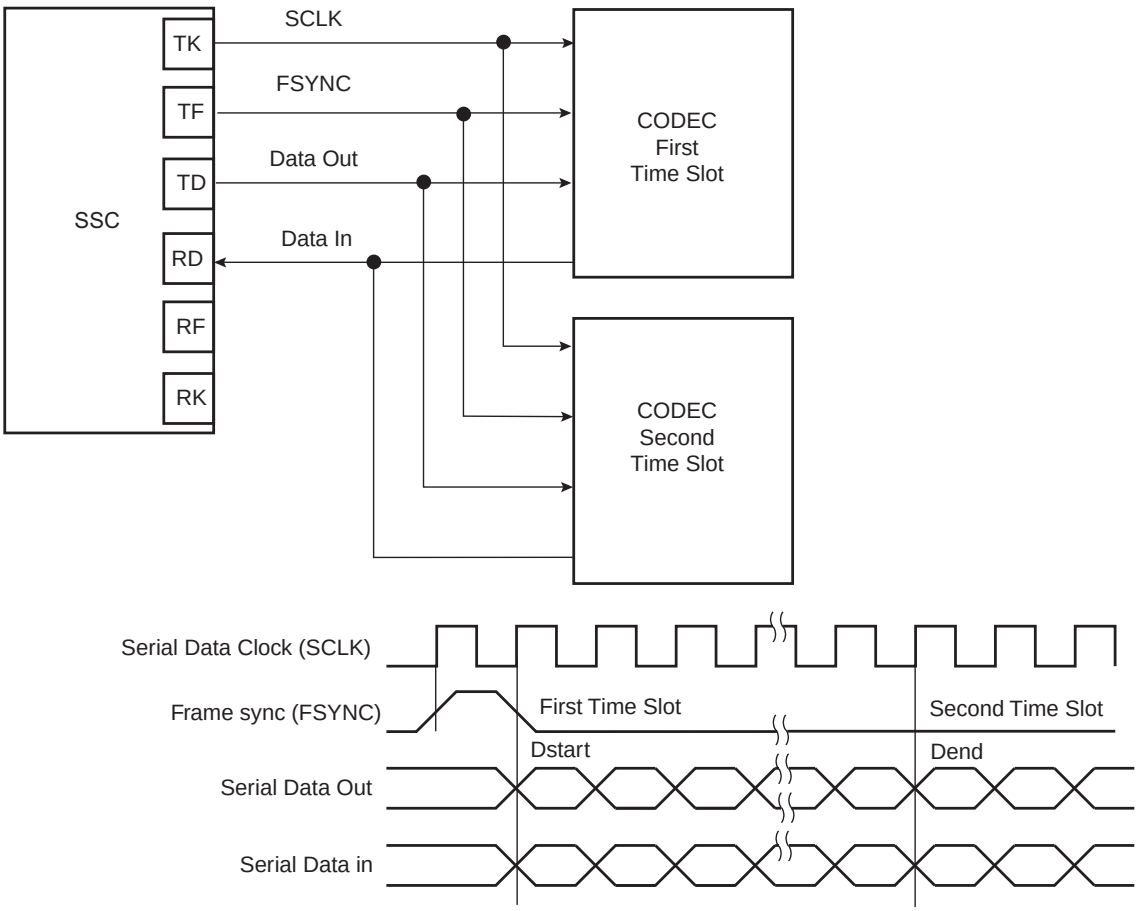


Figure 43-4. Codec Application Block Diagram



Downloaded from Arrow.com.

Figure 43-5. Time Slot Application Block Diagram



43.6 Pin Name List

Table 43-1. I/O Lines Description

Pin Name	Pin Description	Type
RF	Receiver Frame Synchro	Input/Output
RK	Receiver Clock	Input/Output
RD	Receiver Data	Input
TF	Transmitter Frame Synchro	Input/Output
TK	Transmitter Clock	Input/Output
TD	Transmitter Data	Output

Downloaded from Arrow.com.

43.7 Product Dependencies

43.7.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines.

Before using the SSC receiver, the PIO controller must be configured to dedicate the SSC receiver I/O lines to the SSC peripheral mode.

Before using the SSC transmitter, the PIO controller must be configured to dedicate the SSC transmitter I/O lines to the SSC peripheral mode.

Table 43-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
SSC	RD	PA10	C
SSC	RF	PD24	B
SSC	RK	PA22	A
SSC	TD	PB5	D
SSC	TD	PD10	C
SSC	TD	PD26	B
SSC	TF	PB0	D
SSC	TK	PB1	D

43.7.2 Power Management

The SSC is not continuously clocked. The SSC interface may be clocked through the Power Management Controller (PMC), therefore the programmer must first configure the PMC to enable the SSC clock.

43.7.3 Interrupt

The SSC interface has an interrupt line connected to the interrupt controller. Handling interrupts requires programming the interrupt controller before configuring the SSC.

All SSC interrupts can be enabled/disabled configuring the SSC Interrupt Mask Register. Each pending and unmasked SSC interrupt will assert the SSC interrupt line. The SSC interrupt service routine can get the interrupt origin by reading the SSC Interrupt Status Register.

Table 43-3. Peripheral IDs

Instance	ID
SSC	22

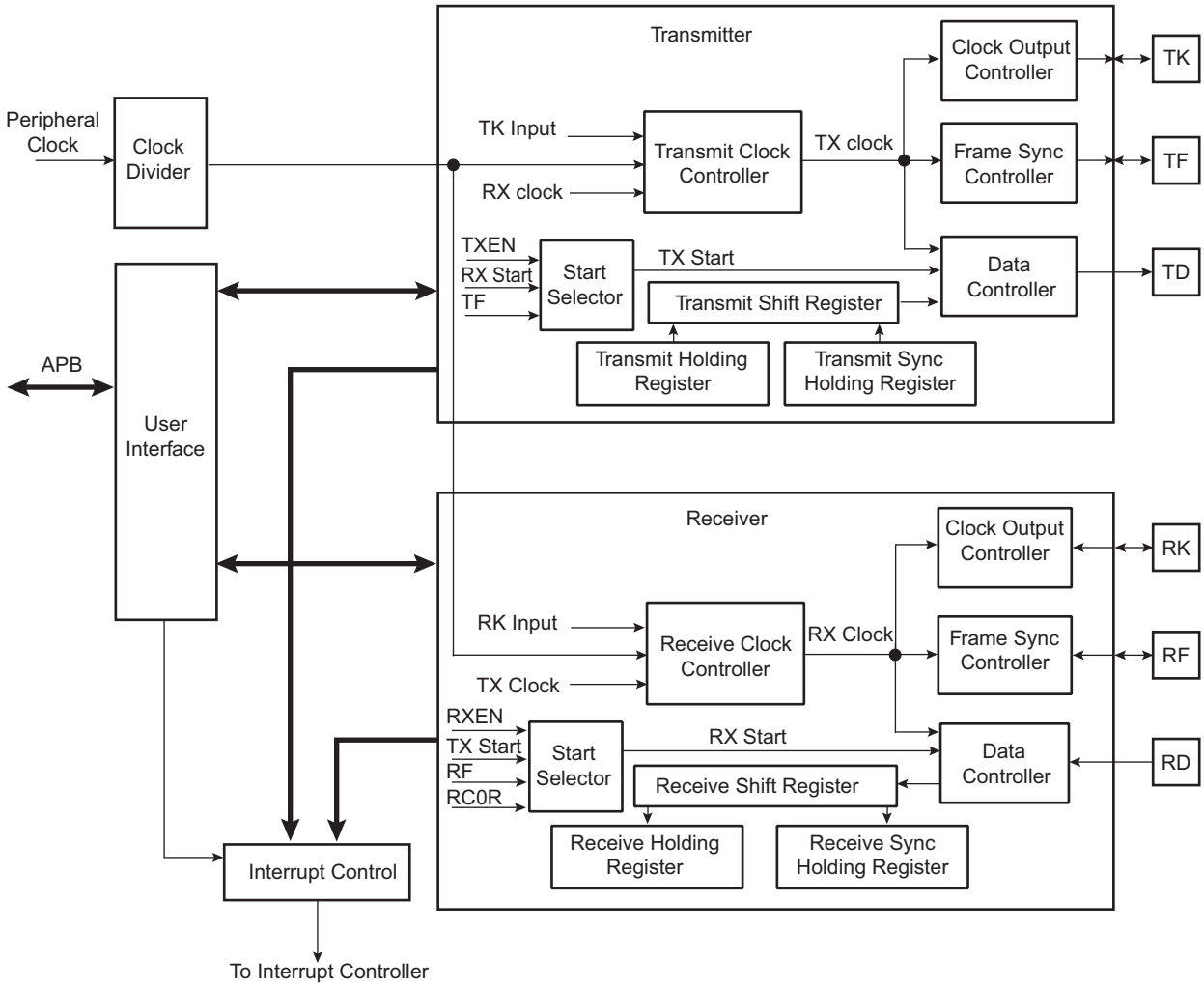
Downloaded from Arrow.com.

43.8 Functional Description

This chapter contains the functional description of the following: SSC Functional Block, Clock Management, Data format, Start, Transmitter, Receiver and Frame Sync.

The receiver and transmitter operate separately. However, they can work synchronously by programming the receiver to use the transmit clock and/or to start a data transfer when transmission starts. Alternatively, this can be done by programming the transmitter to use the receive clock and/or to start a data transfer when reception starts. The transmitter and the receiver can be programmed to operate with the clock signals provided on either the TK or RK pins. This allows the SSC to support many slave-mode data transfers. The maximum clock speed allowed on the TK and RK pins is the peripheral clock divided by 2.

Figure 43-6. SSC Functional Block Diagram



Downloaded from Arrow.com.

43.8.1 Clock Management

The transmitter clock can be generated by:

- an external clock received on the TK I/O pad
- the receiver clock
- the internal clock divider

The receiver clock can be generated by:

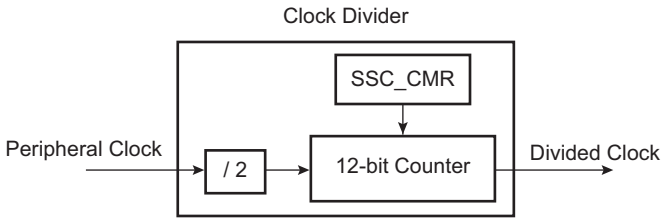
- an external clock received on the RK I/O pad
- the transmitter clock
- the internal clock divider

Furthermore, the transmitter block can generate an external clock on the TK I/O pad, and the receiver block can generate an external clock on the RK I/O pad.

This allows the SSC to support many Master and Slave Mode data transfers.

43.8.1.1 Clock Divider

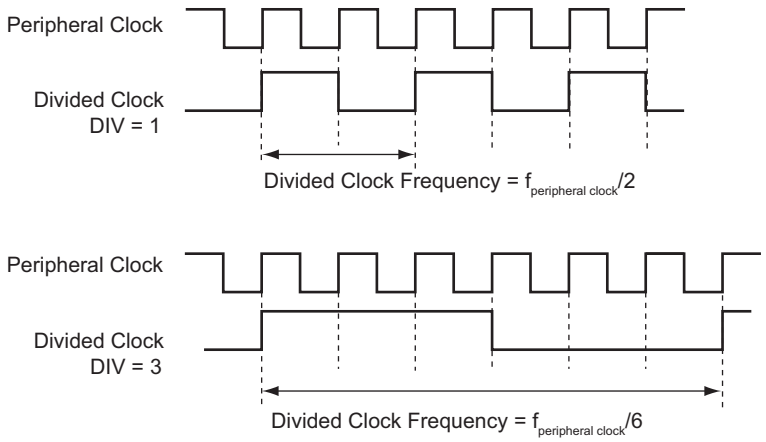
Figure 43-7. Divided Clock Block Diagram



The peripheral clock divider is determined by the 12-bit field DIV counter and comparator (so its maximal value is 4095) in the Clock Mode Register (SSC_CMR), allowing a peripheral clock division by up to 8190. The Divided Clock is provided to both the Receiver and Transmitter. When this field is programmed to 0, the Clock Divider is not used and remains inactive.

When DIV is set to a value equal to or greater than 1, the Divided Clock has a frequency of peripheral clock divided by 2 times DIV. Each level of the Divided Clock has a duration of the peripheral clock multiplied by DIV. This ensures a 50% duty cycle for the Divided Clock regardless of whether the DIV value is even or odd.

Figure 43-8. Divided Clock Generation



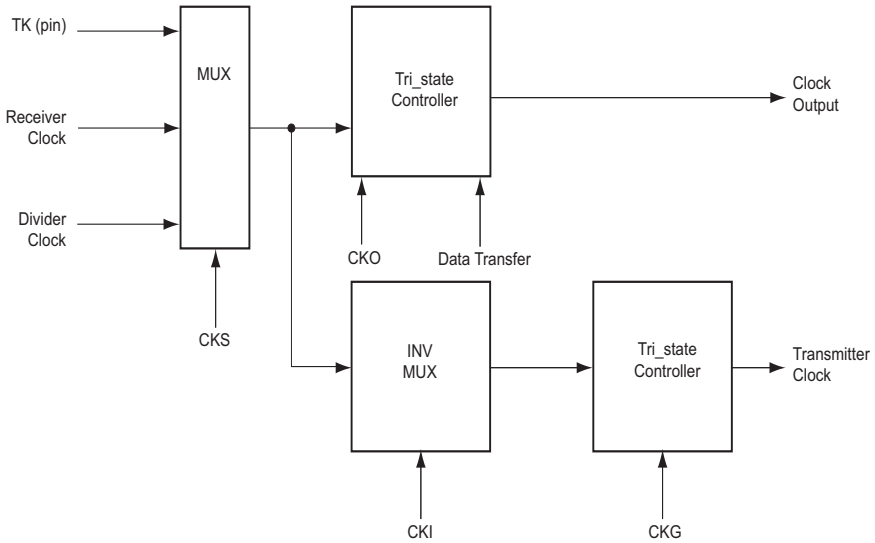
Downloaded from Arrow.com.

43.8.1.2 Transmitter Clock Management

The transmitter clock is generated from the receiver clock or the divider clock or an external clock scanned on the TK I/O pad. The transmitter clock is selected by the CKS field in the Transmit Clock Mode Register (SSC_TCMR). Transmit Clock can be inverted independently by the CKI bits in the SSC_TCMR.

The transmitter can also drive the TK I/O pad continuously or be limited to the actual data transfer. The clock output is configured by the SSC_TCMR. The Transmit Clock Inversion (CKI) bits have no effect on the clock outputs. Programming the SSC_TCMR to select TK pin (CKS field) and at the same time Continuous Transmit Clock (CKO field) can lead to unpredictable results.

Figure 43-9. Transmitter Clock Management



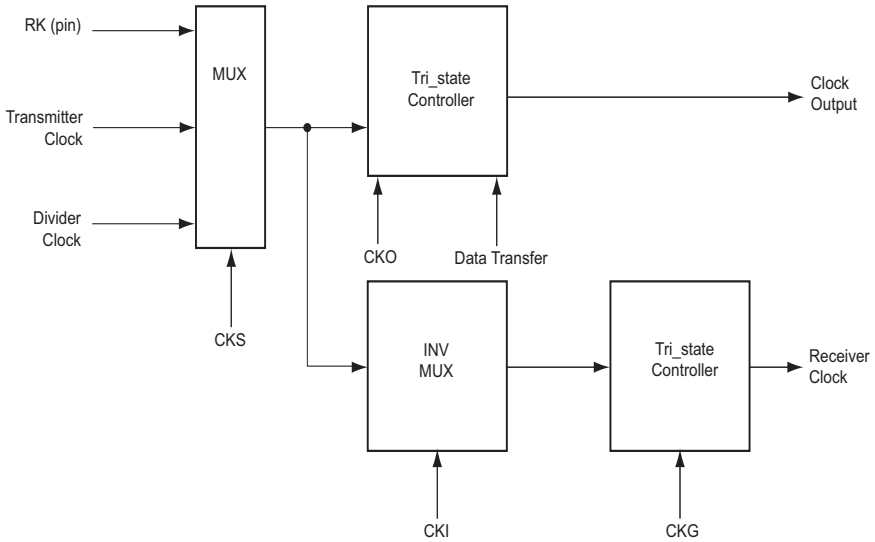
Downloaded from Arrow.com.

43.8.1.3 Receiver Clock Management

The receiver clock is generated from the transmitter clock or the divider clock or an external clock scanned on the RK I/O pad. The Receive Clock is selected by the CKS field in SSC_RCMR (Receive Clock Mode Register). Receive Clocks can be inverted independently by the CKI bits in SSC_RCMR.

The receiver can also drive the RK I/O pad continuously or be limited to the actual data transfer. The clock output is configured by the SSC_RCMR. The Receive Clock Inversion (CKI) bits have no effect on the clock outputs. Programming the SSC_RCMR to select RK pin (CKS field) and at the same time Continuous Receive Clock (CKO field) can lead to unpredictable results.

Figure 43-10. Receiver Clock Management



43.8.1.4 Serial Clock Ratio Considerations

The Transmitter and the Receiver can be programmed to operate with the clock signals provided on either the TK or RK pins. This allows the SSC to support many slave-mode data transfers. In this case, the maximum clock speed allowed on the RK pin is:

- Peripheral clock divided by 2 if Receiver Frame Synchro is input
- Peripheral clock divided by 3 if Receiver Frame Synchro is output

In addition, the maximum clock speed allowed on the TK pin is:

- Peripheral clock divided by 6 if Transmit Frame Synchro is input
- Peripheral clock divided by 2 if Transmit Frame Synchro is output

Downloaded from Arrow.com.

43.8.2 Transmitter Operations

A transmitted frame is triggered by a start event and can be followed by synchronization data before data transmission.

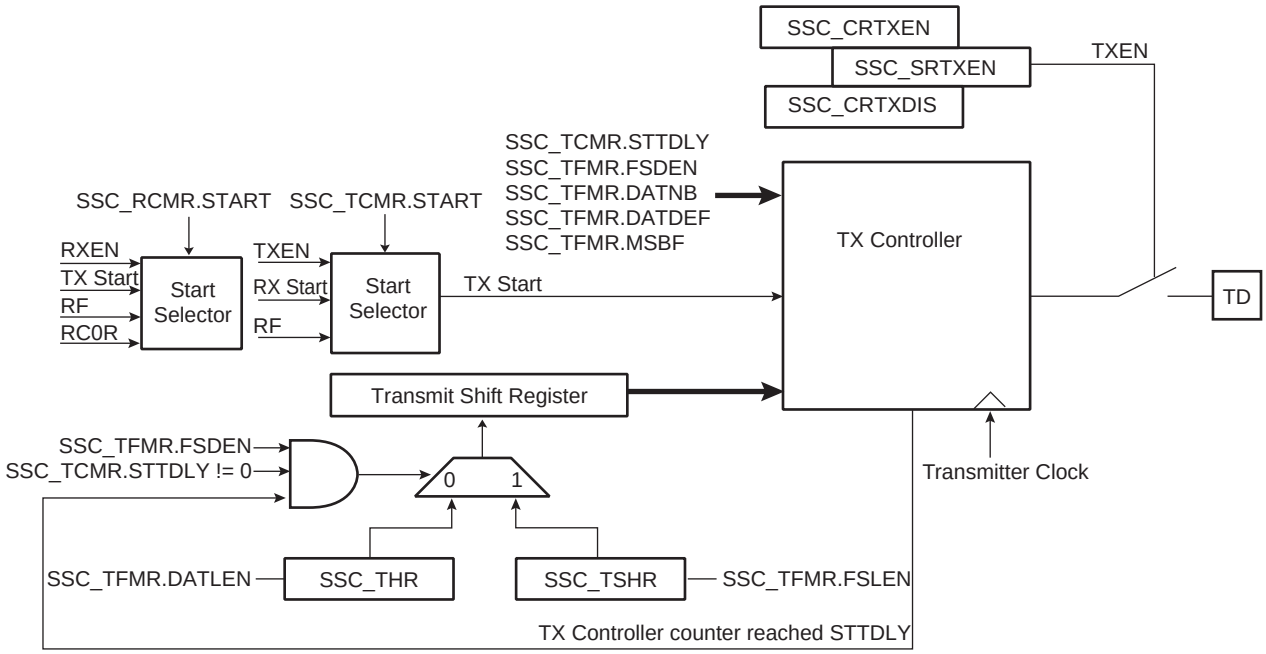
The start event is configured by setting the SSC_TCMR. See Section 43.8.4 “Start” on page 1074.

The frame synchronization is configured setting the Transmit Frame Mode Register (SSC_TFMR). See Section 43.8.5 “Frame Sync” on page 1076.

To transmit data, the transmitter uses a shift register clocked by the transmitter clock signal and the start mode selected in the SSC_TCMR. Data is written by the application to the SSC_THR then transferred to the shift register according to the data format selected.

When both the SSC_THR and the transmit shift register are empty, the status flag TXEMPTY is set in the SSC_SR. When the Transmit Holding register is transferred in the transmit shift register, the status flag TXRDY is set in the SSC_SR and additional data can be loaded in the holding register.

Figure 43-11. Transmitter Block Diagram



Downloaded from Arrow.com.

43.8.3 Receiver Operations

A received frame is triggered by a start event and can be followed by synchronization data before data transmission.

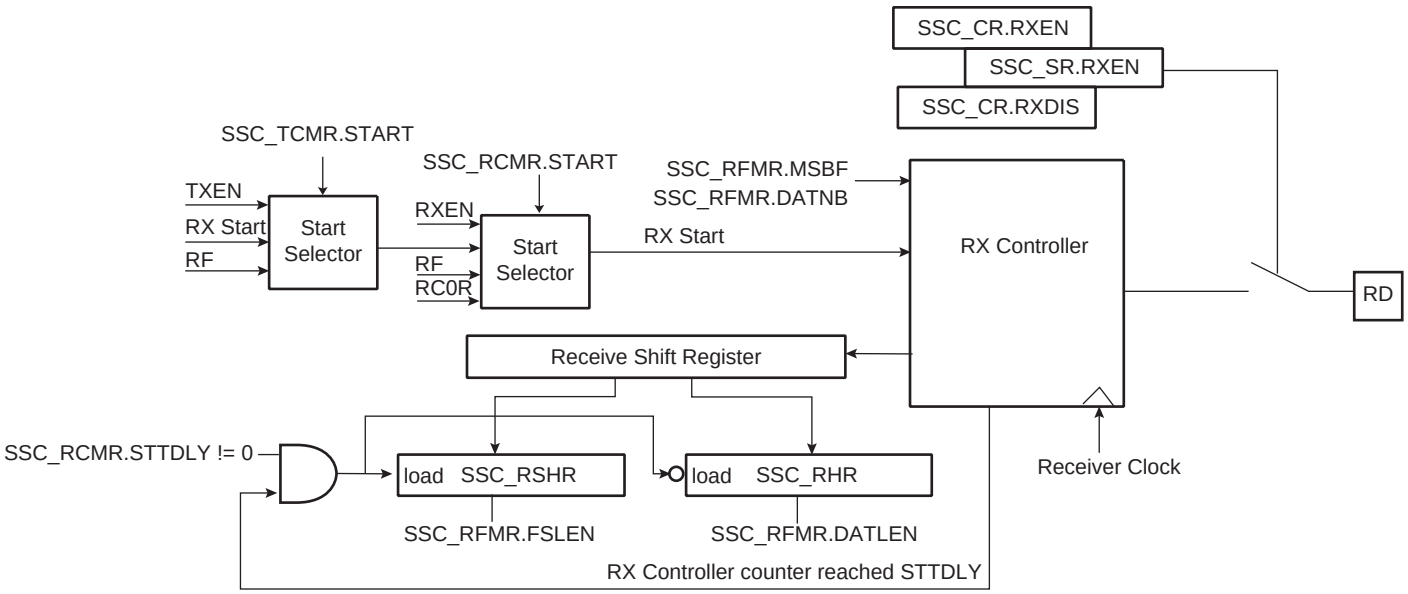
The start event is configured setting the Receive Clock Mode Register (SSC_RCMR). See Section 43.8.4 “Start” on page 1074.

The frame synchronization is configured setting the Receive Frame Mode Register (SSC_RFMR). See Section 43.8.5 “Frame Sync” on page 1076.

The receiver uses a shift register clocked by the receiver clock signal and the start mode selected in the SSC_RCMR. The data is transferred from the shift register depending on the data format selected.

When the receiver shift register is full, the SSC transfers this data in the holding register, the status flag RXRDY is set in the SSC_SR and the data can be read in the receiver holding register. If another transfer occurs before read of the Receive Holding Register (SSC_RHR), the status flag OVERUN is set in the SSC_SR and the receiver shift register is transferred in the SSC_RHR.

Figure 43-12. Receiver Block Diagram



Downloaded from Arrow.com.

43.8.4 Start

The transmitter and receiver can both be programmed to start their operations when an event occurs, respectively in the Transmit Start Selection (START) field of SSC_TCMR and in the Receive Start Selection (START) field of SSC_RCMR.

Under the following conditions the start event is independently programmable:

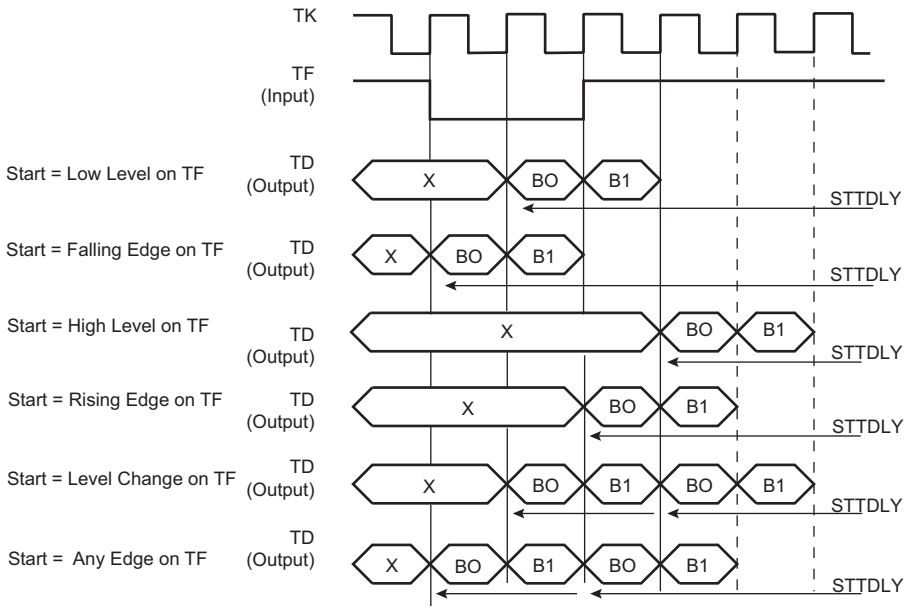
- Continuous. In this case, the transmission starts as soon as a word is written in SSC_THR and the reception starts as soon as the Receiver is enabled.
- Synchronously with the transmitter/receiver
- On detection of a falling/rising edge on TF/RF
- On detection of a low level/high level on TF/RF
- On detection of a level change or an edge on TF/RF

A start can be programmed in the same manner on either side of the Transmit/Receive Clock Register (SSC_RCMR/SSC_TCMR). Thus, the start could be on TF (Transmit) or RF (Receive).

Moreover, the Receiver can start when data is detected in the bit stream with the Compare Functions.

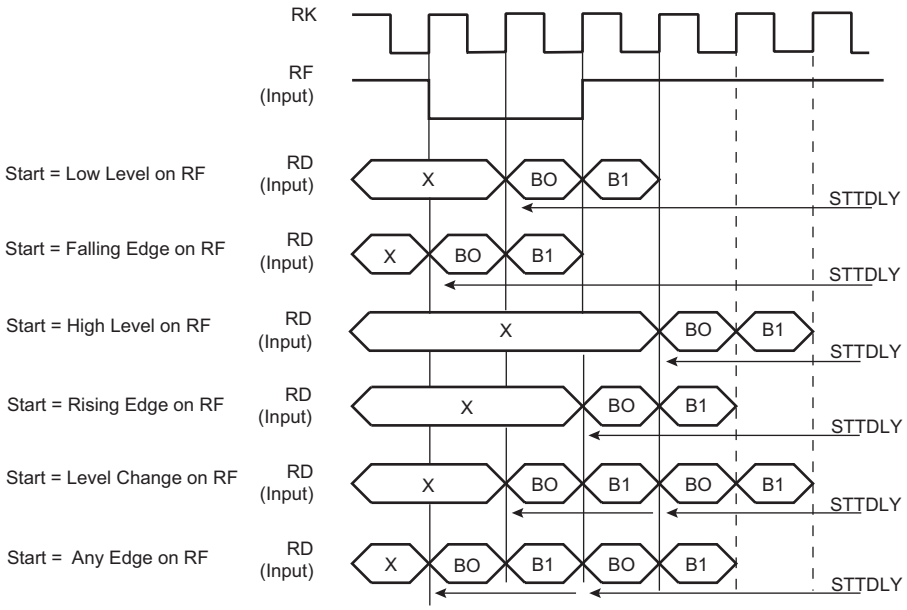
Detection on TF/RF input/output is done by the field FSOS of the Transmit/Receive Frame Mode Register (SSC_TFMR/SSC_RFMR).

Figure 43-13. Transmit Start Mode



Downloaded from Arrow.com.

Figure 43-14. Receive Pulse/Edge Start Modes



Downloaded from Arrow.com.

43.8.5 Frame Sync

The Transmitter and Receiver Frame Sync pins, TF and RF, can be programmed to generate different kinds of frame synchronization signals. The Frame Sync Output Selection (FSOS) field in the Receive Frame Mode Register (SSC_RFMR) and in the Transmit Frame Mode Register (SSC_TFMR) are used to select the required waveform.

- Programmable low or high levels during data transfer are supported.
- Programmable high levels before the start of data transfers or toggling are also supported.

If a pulse waveform is selected, the Frame Sync Length (FSLEN) field in SSC_RFMR and SSC_TFMR programs the length of the pulse, from 1 bit time up to 256 bit times.

The periodicity of the Receive and Transmit Frame Sync pulse output can be programmed through the Period Divider Selection (PERIOD) field in SSC_RCMR and SSC_TCMR.

43.8.5.1 Frame Sync Data

Frame Sync Data transmits or receives a specific tag during the Frame Sync signal.

During the Frame Sync signal, the Receiver can sample the RD line and store the data in the Receive Sync Holding Register and the transmitter can transfer Transmit Sync Holding Register in the shift register. The data length to be sampled/shifted out during the Frame Sync signal is programmed by the FSLEN field in SSC_RFMR/SSC_TFMR and has a maximum value of 256.

Concerning the Receive Frame Sync Data operation, if the Frame Sync Length is equal to or lower than the delay between the start event and the actual data reception, the data sampling operation is performed in the Receive Sync Holding Register through the receive shift register.

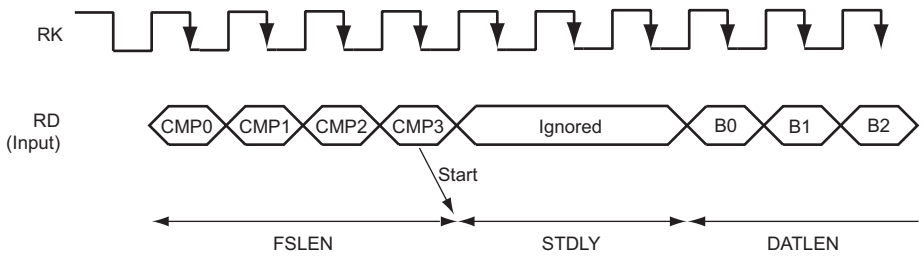
The Transmit Frame Sync Operation is performed by the transmitter only if the bit Frame Sync Data Enable (FSDEN) in SSC_TFMR is set. If the Frame Sync length is equal to or lower than the delay between the start event and the actual data transmission, the normal transmission has priority and the data contained in the Transmit Sync Holding Register is transferred in the Transmit Register, then shifted out.

43.8.5.2 Frame Sync Edge Detection

The Frame Sync Edge detection is programmed by the FSEDGE field in SSC_RFMR/SSC_TFMR. This sets the corresponding flags RXSYN/TXSYN in the SSC Status Register (SSC_SR) on frame synchro edge detection (signals RF/TF).

43.8.6 Receive Compare Modes

Figure 43-15. Receive Compare Modes



43.8.6.1 Compare Functions

The length of the comparison patterns (Compare 0, Compare 1) and thus the number of bits they are compared to is defined by FSLEN, but with a maximum value of 256 bits. Comparison is always done by comparing the last bits received with the comparison pattern. Compare 0 can be one start event of the Receiver. In this case, the receiver compares at each new sample the last bits received at the Compare 0 pattern contained in the Compare 0 Register (SSC_RC0R). When this start event is selected, the user can program the Receiver to start a new data

Downloaded from Arrow.com.

transfer either by writing a new Compare 0, or by receiving continuously until Compare 1 occurs. This selection is done with the STOP bit in the SSC_RCMR.

43.8.7 Data Format

The data framing format of both the transmitter and the receiver are programmable through the Transmitter Frame Mode Register (SSC_TFMR) and the Receiver Frame Mode Register (SSC_RFMR). In either case, the user can independently select the following parameters:

- Event that starts the data transfer (START)
- Delay in number of bit periods between the start event and the first data bit (STTDLY)
- Length of the data (DATLEN)
- Number of data to be transferred for each start event (DATNB)
- Length of synchronization transferred for each start event (FSLEN)
- Bit sense: most or lowest significant bit first (MSBF)

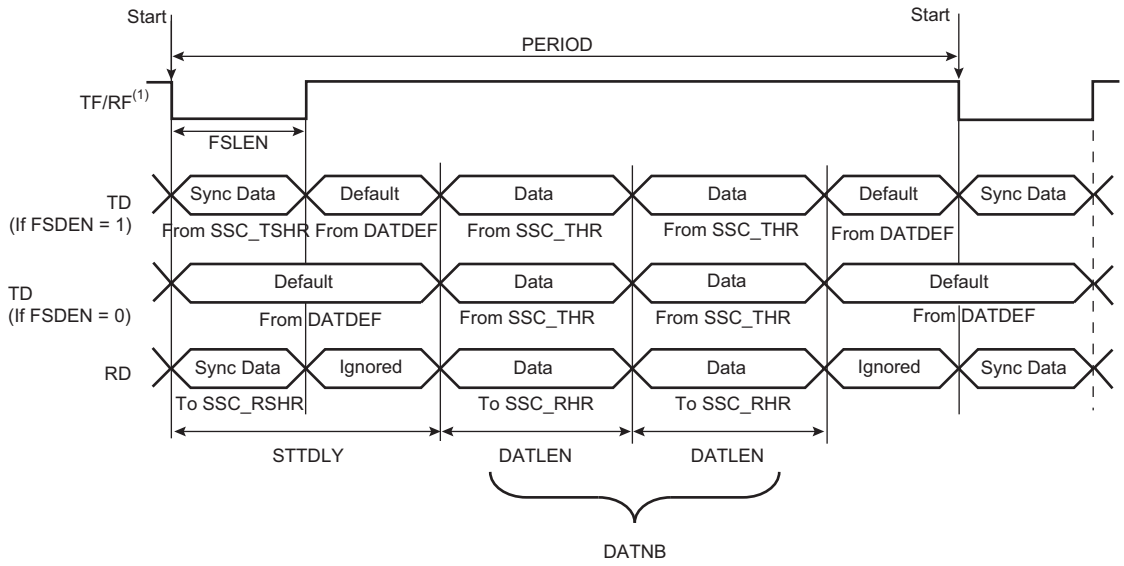
Additionally, the transmitter can be used to transfer synchronization and select the level driven on the TD pin while not in data transfer operation. This is done respectively by the Frame Sync Data Enable (FSDEN) and by the Data Default Value (DATDEF) bits in SSC_TFMR.

Table 43-4. Data Frame Registers

Transmitter	Receiver	Field	Length	Comment
SSC_TFMR	SSC_RFMR	DATLEN	Up to 32	Size of word
SSC_TFMR	SSC_RFMR	DATNB	Up to 16	Number of words transmitted in frame
SSC_TFMR	SSC_RFMR	MSBF	–	Most significant bit first
SSC_TFMR	SSC_RFMR	FSLEN	Up to 256	Size of Synchro data register
SSC_TFMR	–	DATDEF	0 or 1	Data default value ended
SSC_TFMR	–	FSDEN	–	Enable send SSC_TSHR
SSC_TCMR	SSC_RCMR	PERIOD	Up to 512	Frame size
SSC_TCMR	SSC_RCMR	STTDLY	Up to 255	Size of transmit start delay

Downloaded from Arrow.com.

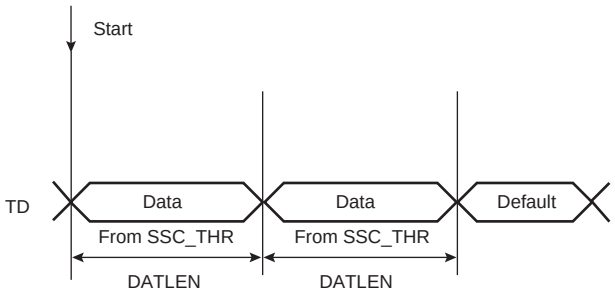
Figure 43-16. Transmit and Receive Frame Format in Edge/Pulse Start Modes



Note: 1. Example of input on falling edge of TF/RF.

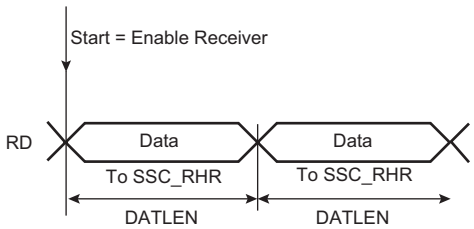
In the example illustrated in Figure 43-17 “Transmit Frame Format in Continuous Mode (STTDLY = 0)”, the SSC_THR is loaded twice. The FSDEN value has no effect on the transmission. SyncData cannot be output in continuous mode.

Figure 43-17. Transmit Frame Format in Continuous Mode (STTDLY = 0)



Start: 1. TXEMPTY set to 1
2. Write into the SSC_THR

Figure 43-18. Receive Frame Format in Continuous Mode (STTDLY = 0)



Downloaded from Arrow.com.

43.8.8 Loop Mode

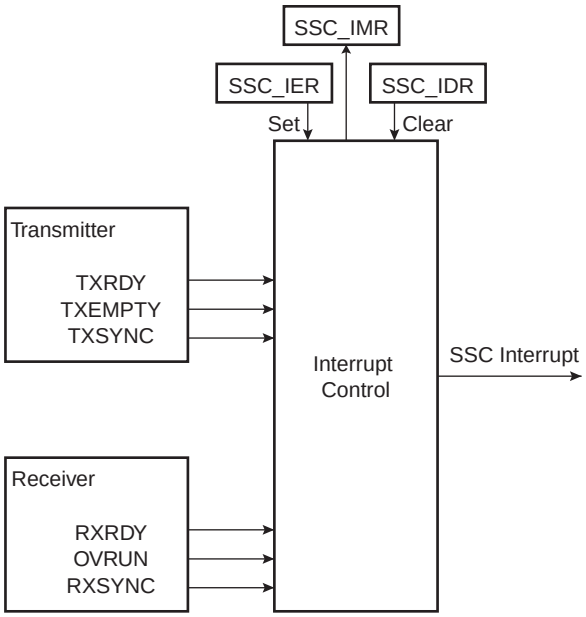
The receiver can be programmed to receive transmissions from the transmitter. This is done by setting the Loop Mode (LOOP) bit in the SSC_RFMR. In this case, RD is connected to TD, RF is connected to TF and RK is connected to TK.

43.8.9 Interrupt

Most bits in the SSC_SR have a corresponding bit in interrupt management registers.

The SSC can be programmed to generate an interrupt when it detects an event. The interrupt is controlled by writing the Interrupt Enable Register (SSC_IER) and Interrupt Disable Register (SSC_IDR). These registers enable and disable, respectively, the corresponding interrupt by setting and clearing the corresponding bit in the Interrupt Mask Register (SSC_IMR), which controls the generation of interrupts by asserting the SSC interrupt line connected to the interrupt controller.

Figure 43-19. Interrupt Block Diagram



Downloaded from Arrow.com.

43.8.10 Register Write Protection

To prevent any single software error from corrupting AIC behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the SSC Write Protection Mode Register (SSC_WPMR).

If a write access to a write-protected register is detected, the WPVS flag in the SSC Write Protection Status Register (SSC_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading the SSC_WPSR.

The following registers can be write-protected:

- SSC Clock Mode Register
- SSC Receive Clock Mode Register
- SSC Receive Frame Mode Register
- SSC Transmit Clock Mode Register
- SSC Transmit Frame Mode Register
- SSC Receive Compare 0 Register
- SSC Receive Compare 1 Register

Downloaded from Arrow.com.

43.9 Synchronous Serial Controller (SSC) User Interface

Table 43-5. Register Mapping

Offset	Register	Name	Access	Reset
0x0	Control Register	SSC_CR	Write-only	–
0x4	Clock Mode Register	SSC_CMR	Read/Write	0x0
0x8–0xC	Reserved	–	–	–
0x10	Receive Clock Mode Register	SSC_RCMR	Read/Write	0x0
0x14	Receive Frame Mode Register	SSC_RFMR	Read/Write	0x0
0x18	Transmit Clock Mode Register	SSC_TCMR	Read/Write	0x0
0x1C	Transmit Frame Mode Register	SSC_TFMR	Read/Write	0x0
0x20	Receive Holding Register	SSC_RHR	Read-only	0x0
0x24	Transmit Holding Register	SSC_THR	Write-only	–
0x28–0x2C	Reserved	–	–	–
0x30	Receive Sync. Holding Register	SSC_RSHR	Read-only	0x0
0x34	Transmit Sync. Holding Register	SSC_TSHR	Read/Write	0x0
0x38	Receive Compare 0 Register	SSC_RC0R	Read/Write	0x0
0x3C	Receive Compare 1 Register	SSC_RC1R	Read/Write	0x0
0x40	Status Register	SSC_SR	Read-only	0x000000CC
0x44	Interrupt Enable Register	SSC_IER	Write-only	–
0x48	Interrupt Disable Register	SSC_IDR	Write-only	–
0x4C	Interrupt Mask Register	SSC_IMR	Read-only	0x0
0x50–0xE0	Reserved	–	–	–
0xE4	Write Protection Mode Register	SSC_WPMR	Read/Write	0x0
0xE8	Write Protection Status Register	SSC_WPSR	Read-only	0x0
0xEC–0xFC	Reserved	–	–	–
0x100–0x124	Reserved	–	–	–

Downloaded from Arrow.com.

43.9.1 SSC Control Register

Name: SSC_CR

Address: 0x40004000

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
SWRST	–	–	–	–	–	TXDIS	TXEN
7	6	5	4	3	2	1	0
–	–	–	–	–	–	RXDIS	RXEN

- **RXEN: Receive Enable**

0: No effect.

1: Enables Receive if RXDIS is not set.

- **RXDIS: Receive Disable**

0: No effect.

1: Disables Receive. If a character is currently being received, disables at end of current character reception.

- **TXEN: Transmit Enable**

0: No effect.

1: Enables Transmit if TXDIS is not set.

- **TXDIS: Transmit Disable**

0: No effect.

1: Disables Transmit. If a character is currently being transmitted, disables at end of current character transmission.

- **SWRST: Software Reset**

0: No effect.

1: Performs a software reset. Has priority on any other bit in SSC_CR.

Downloaded from Arrow.com.

43.9.2 SSC Clock Mode Register

Name: SSC_CMR

Address: 0x40004004

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	DIV			

7	6	5	4	3	2	1	0
DIV							

This register can only be written if the WPEN bit is cleared in the SSC Write Protection Mode Register.

- **DIV: Clock Divider**

0: The Clock Divider is not active.

Any other value: The divided clock equals the peripheral clock divided by 2 times DIV.
The maximum bit rate is $f_{peripheral\ clock}/2$. The minimum bit rate is $f_{peripheral\ clock}/2 \times 4095 = f_{peripheral\ clock}/8190$.

Downloaded from Arrow.com.

43.9.3 SSC Receive Clock Mode Register

Name: SSC_RCMR

Address: 0x40004010

Access: Read/Write

31	30	29	28	27	26	25	24
PERIOD							

23	22	21	20	19	18	17	16
STTDLY							

15	14	13	12	11	10	9	8
–	–	–	STOP	START			

7	6	5	4	3	2	1	0
CKG		CKI	CKO			CKS	

This register can only be written if the WPEN bit is cleared in the SSC Write Protection Mode Register.

- **CKS: Receive Clock Selection**

Value	Name	Description
0	MCK	Divided Clock
1	TK	TK Clock signal
2	RK	RK pin

- **CKO: Receive Clock Output Mode Selection**

Value	Name	Description
0	NONE	None, RK pin is an input
1	CONTINUOUS	Continuous Receive Clock, RK pin is an output
2	TRANSFER	Receive Clock only during data transfers, RK pin is an output

- **CKI: Receive Clock Inversion**

0: The data inputs (Data and Frame Sync signals) are sampled on Receive Clock falling edge. The Frame Sync signal output is shifted out on Receive Clock rising edge.

1: The data inputs (Data and Frame Sync signals) are sampled on Receive Clock rising edge. The Frame Sync signal output is shifted out on Receive Clock falling edge.

CKI affects only the Receive Clock and not the output clock signal.

Downloaded from Arrow.com.

- **CKG: Receive Clock Gating Selection**

Value	Name	Description
0	CONTINUOUS	None
1	EN_RF_LOW	Receive Clock enabled only if RF Low
2	EN_RF_HIGH	Receive Clock enabled only if RF High

- **START: Receive Start Selection**

Value	Name	Description
0	CONTINUOUS	Continuous, as soon as the receiver is enabled, and immediately after the end of transfer of the previous data.
1	TRANSMIT	Transmit start
2	RF_LOW	Detection of a low level on RF signal
3	RF_HIGH	Detection of a high level on RF signal
4	RF_FALLING	Detection of a falling edge on RF signal
5	RF_RISING	Detection of a rising edge on RF signal
6	RF_LEVEL	Detection of any level change on RF signal
7	RF_EDGE	Detection of any edge on RF signal
8	CMP_0	Compare 0

- **STOP: Receive Stop Selection**

0: After completion of a data transfer when starting with a Compare 0, the receiver stops the data transfer and waits for a new compare 0.

1: After starting a receive with a Compare 0, the receiver operates in a continuous mode until a Compare 1 is detected.

- **STTDLY: Receive Start Delay**

If STTDLY is not 0, a delay of STTDLY clock cycles is inserted between the start event and the actual start of reception. When the Receiver is programmed to start synchronously with the Transmitter, the delay is also applied.

Note: It is very important that STTDLY be set carefully. If STTDLY must be set, it should be done in relation to TAG (Receive Sync Data) reception.

- **PERIOD: Receive Period Divider Selection**

This field selects the divider to apply to the selected Receive Clock in order to generate a new Frame Sync Signal. If 0, no PERIOD signal is generated. If not 0, a PERIOD signal is generated each 2 x (PERIOD + 1) Receive Clock.

Downloaded from Arrow.com.

43.9.4 SSC Receive Frame Mode Register

Name: SSC_RFMR

Address: 0x40004014

Access: Read/Write

31	30	29	28	27	26	25	24
FSLEN_EXT				–	–	–	FSEDGE

23	22	21	20	19	18	17	16
–	FSOS			FSLEN			

15	14	13	12	11	10	9	8
–	–	–	–	DATNB			

7	6	5	4	3	2	1	0
MSBF	–	LOOP	DATLEN				

This register can only be written if the WPEN bit is cleared in the SSC Write Protection Mode Register.

- **DATLEN: Data Length**

0: Forbidden value (1-bit data length not supported).

Any other value: The bit stream contains DATLEN + 1 data bits.

- **LOOP: Loop Mode**

0: Normal operating mode.

1: RD is driven by TD, RF is driven by TF and TK drives RK.

- **MSBF: Most Significant Bit First**

0: The lowest significant bit of the data register is sampled first in the bit stream.

1: The most significant bit of the data register is sampled first in the bit stream.

- **DATNB: Data Number per Frame**

This field defines the number of data words to be received after each transfer start, which is equal to (DATNB + 1).

- **FSLEN: Receive Frame Sync Length**

This field defines the number of bits sampled and stored in the Receive Sync Data Register. When this mode is selected by the START field in the Receive Clock Mode Register, it also determines the length of the sampled data to be compared to the Compare 0 or Compare 1 register.

This field is used with FSLEN_EXT to determine the pulse length of the Receive Frame Sync signal.

Pulse length is equal to FSLEN + (FSLEN_EXT × 16) + 1 Receive Clock periods.

Downloaded from Arrow.com.

- **FSOS: Receive Frame Sync Output Selection**

Value	Name	Description
0	NONE	None, RF pin is an input
1	NEGATIVE	Negative Pulse, RF pin is an output
2	POSITIVE	Positive Pulse, RF pin is an output
3	LOW	Driven Low during data transfer, RF pin is an output
4	HIGH	Driven High during data transfer, RF pin is an output
5	TOGGLING	Toggling at each start of data transfer, RF pin is an output

- **FSEDGE: Frame Sync Edge Detection**

Determines which edge on Frame Sync will generate the interrupt RXSYN in the SSC Status Register.

Value	Name	Description
0	POSITIVE	Positive Edge Detection
1	NEGATIVE	Negative Edge Detection

- **FSLEN_EXT: FSLEN Field Extension**

Extends FSLEN field. For details, refer to FSLEN bit description on page 1086.

Downloaded from Arrow.com.

43.9.5 SSC Transmit Clock Mode Register

Name: SSC_TCMR

Address: 0x40004018

Access: Read/Write

31	30	29	28	27	26	25	24
PERIOD							

23	22	21	20	19	18	17	16
STTDLY							

15	14	13	12	11	10	9	8
–	–	–	–	START			

7	6	5	4	3	2	1	0
CKG		CKI	CKO			CKS	

This register can only be written if the WPEN bit is cleared in the SSC Write Protection Mode Register.

- **CKS: Transmit Clock Selection**

Value	Name	Description
0	MCK	Divided Clock
1	RK	RK Clock signal
2	TK	TK pin

- **CKO: Transmit Clock Output Mode Selection**

Value	Name	Description
0	NONE	None, TK pin is an input
1	CONTINUOUS	Continuous Transmit Clock, TK pin is an output
2	TRANSFER	Transmit Clock only during data transfers, TK pin is an output

- **CKI: Transmit Clock Inversion**

0: The data outputs (Data and Frame Sync signals) are shifted out on Transmit Clock falling edge. The Frame sync signal input is sampled on Transmit clock rising edge.

1: The data outputs (Data and Frame Sync signals) are shifted out on Transmit Clock rising edge. The Frame sync signal input is sampled on Transmit clock falling edge.

CKI affects only the Transmit Clock and not the output clock signal.

- **CKG: Transmit Clock Gating Selection**

Value	Name	Description
0	CONTINUOUS	None
1	EN_TF_LOW	Transmit Clock enabled only if TF Low
2	EN_TF_HIGH	Transmit Clock enabled only if TF High

Downloaded from Arrow.com.

- **START: Transmit Start Selection**

Value	Name	Description
0	CONTINUOUS	Continuous, as soon as a word is written in the SSC_THR (if Transmit is enabled), and immediately after the end of transfer of the previous data
1	RECEIVE	Receive start
2	TF_LOW	Detection of a low level on TF signal
3	TF_HIGH	Detection of a high level on TF signal
4	TF_FALLING	Detection of a falling edge on TF signal
5	TF_RISING	Detection of a rising edge on TF signal
6	TF_LEVEL	Detection of any level change on TF signal
7	TF_EDGE	Detection of any edge on TF signal

- **STTDLY: Transmit Start Delay**

If STTDLY is not 0, a delay of STTDLY clock cycles is inserted between the start event and the actual start of transmission of data. When the Transmitter is programmed to start synchronously with the Receiver, the delay is also applied.

Note: Note: STTDLY must be set carefully. If STTDLY is too short in respect to TAG (Transmit Sync Data) emission, data is emitted instead of the end of TAG.

- **PERIOD: Transmit Period Divider Selection**

This field selects the divider to apply to the selected Transmit Clock to generate a new Frame Sync Signal. If 0, no period signal is generated. If not 0, a period signal is generated at each 2 × (PERIOD + 1) Transmit Clock.

Downloaded from Arrow.com.

43.9.6 SSC Transmit Frame Mode Register

Name: SSC_TFMR

Address: 0x4000401C

Access: Read/Write

31	30	29	28	27	26	25	24
FSLEN_EXT				–	–	–	FSEDGE

23	22	21	20	19	18	17	16
FSDEN	FSOS			FSLEN			

15	14	13	12	11	10	9	8
–	–	–	–	DATNB			

7	6	5	4	3	2	1	0
MSBF	–	DATDEF	DATLEN				

This register can only be written if the WPEN bit is cleared in the SSC Write Protection Mode Register.

- **DATLEN: Data Length**

0: Forbidden value (1-bit data length not supported).

Any other value: The bit stream contains DATLEN + 1 data bits. .

- **DATDEF: Data Default Value**

This bit defines the level driven on the TD pin while out of transmission. Note that if the pin is defined as multi-drive by the PIO Controller, the pin is enabled only if the SCC TD output is 1.

- **MSBF: Most Significant Bit First**

0: The lowest significant bit of the data register is shifted out first in the bit stream.

1: The most significant bit of the data register is shifted out first in the bit stream.

- **DATNB: Data Number per Frame**

This field defines the number of data words to be transferred after each transfer start, which is equal to (DATNB + 1).

- **FSLEN: Transmit Frame Sync Length**

This field defines the length of the Transmit Frame Sync signal and the number of bits shifted out from the Transmit Sync Data Register if FSDEN is 1.

This field is used with FSLEN_EXT to determine the pulse length of the Transmit Frame Sync signal.

Pulse length is equal to FSLEN + (FSLEN_EXT $\times$ 16) + 1 Transmit Clock period.

Downloaded from Arrow.com.

- **FSOS: Transmit Frame Sync Output Selection**

Value	Name	Description
0	NONE	None, TF pin is an input
1	NEGATIVE	Negative Pulse, TF pin is an output
2	POSITIVE	Positive Pulse, TF pin is an output
3	LOW	Driven Low during data transfer
4	HIGH	Driven High during data transfer
5	TOGGLING	Toggling at each start of data transfer

- **FSDEN: Frame Sync Data Enable**

0: The TD line is driven with the default value during the Transmit Frame Sync signal.

1: SSC_TSHR value is shifted out during the transmission of the Transmit Frame Sync signal.

- **FSEDGE: Frame Sync Edge Detection**

Determines which edge on frame sync will generate the interrupt TXSYN (Status Register).

Value	Name	Description
0	POSITIVE	Positive Edge Detection
1	NEGATIVE	Negative Edge Detection

- **FSLEN_EXT: FSLEN Field Extension**

Extends FSLEN field. For details, refer to FSLEN bit description on page 1090.

Downloaded from Arrow.com.

43.9.7 SSC Receive Holding Register

Name: SSC_RHR

Address: 0x40004020

Access: Read-only

31	30	29	28	27	26	25	24
RDAT							

23	22	21	20	19	18	17	16
RDAT							

15	14	13	12	11	10	9	8
RDAT							

7	6	5	4	3	2	1	0
RDAT							

- **RDAT: Receive Data**

Right aligned regardless of the number of data bits defined by DATLEN in SSC_RFMR.

Downloaded from Arrow.com.

43.9.8 SSC Transmit Holding Register

Name: SSC_THR

Address: 0x40004024

Access: Write-only

31	30	29	28	27	26	25	24
TDAT							

23	22	21	20	19	18	17	16
TDAT							

15	14	13	12	11	10	9	8
TDAT							

7	6	5	4	3	2	1	0
TDAT							

- **TDAT: Transmit Data**

Right aligned regardless of the number of data bits defined by DATLEN in SSC_TFMR.

Downloaded from Arrow.com.

43.9.9 SSC Receive Synchronization Holding Register

Name: SSC_RSHR

Address: 0x40004030

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
RSDAT							

7	6	5	4	3	2	1	0
RSDAT							

- **RSDAT: Receive Synchronization Data**

Downloaded from Arrow.com.

43.9.10 SSC Transmit Synchronization Holding Register

Name: SSC_TSHR

Address: 0x40004034

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
TSDAT							

7	6	5	4	3	2	1	0
TSDAT							

- **TSDAT: Transmit Synchronization Data**

Downloaded from Arrow.com.

43.9.11 SSC Receive Compare 0 Register

Name: SSC_RC0R

Address: 0x40004038

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
CP0							

7	6	5	4	3	2	1	0
CP0							

This register can only be written if the WPEN bit is cleared in the SSC Write Protection Mode Register.

- **CP0: Receive Compare Data 0**

Downloaded from Arrow.com.

43.9.12 SSC Receive Compare 1 Register

Name: SSC_RC1R

Address: 0x4000403C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
CP1							

7	6	5	4	3	2	1	0
CP1							

This register can only be written if the WPEN bit is cleared in the SSC Write Protection Mode Register.

- **CP1: Receive Compare Data 1**

Downloaded from Arrow.com.

43.9.13 SSC Status Register

Name: SSC_SR

Address: 0x40004040

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	RXEN	TXEN
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
–	–	OVRUN	RXRDY	–	–	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready**

0: Data has been loaded in SSC_THR and is waiting to be loaded in the transmit shift register (TSR).

1: SSC_THR is empty.

- **TXEMPTY: Transmit Empty**

0: Data remains in SSC_THR or is currently transmitted from TSR.

1: Last data written in SSC_THR has been loaded in TSR and last data loaded in TSR has been transmitted.

- **RXRDY: Receive Ready**

0: SSC_RHR is empty.

1: Data has been received and loaded in SSC_RHR.

- **OVRUN: Receive Overrun**

0: No data has been loaded in SSC_RHR while previous data has not been read since the last read of the Status Register.

1: Data has been loaded in SSC_RHR while previous data has not yet been read since the last read of the Status Register.

- **CP0: Compare 0**

0: A compare 0 has not occurred since the last read of the Status Register.

1: A compare 0 has occurred since the last read of the Status Register.

- **CP1: Compare 1**

0: A compare 1 has not occurred since the last read of the Status Register.

1: A compare 1 has occurred since the last read of the Status Register.

- **TXSYN: Transmit Sync**

0: A Tx Sync has not occurred since the last read of the Status Register.

1: A Tx Sync has occurred since the last read of the Status Register.

Downloaded from Arrow.com.

- **RXSYN: Receive Sync**

0: An Rx Sync has not occurred since the last read of the Status Register.

1: An Rx Sync has occurred since the last read of the Status Register.

- **TXEN: Transmit Enable**

0: Transmit is disabled.

1: Transmit is enabled.

- **RXEN: Receive Enable**

0: Receive is disabled.

1: Receive is enabled.

Downloaded from Arrow.com.

43.9.14 SSC Interrupt Enable Register

Name: SSC_IER

Address: 0x40004044

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
–	–	OVRUN	RXRDY	–	–	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready Interrupt Enable**

0: No effect.

1: Enables the Transmit Ready Interrupt.

- **TXEMPTY: Transmit Empty Interrupt Enable**

0: No effect.

1: Enables the Transmit Empty Interrupt.

- **RXRDY: Receive Ready Interrupt Enable**

0: No effect.

1: Enables the Receive Ready Interrupt.

- **OVRUN: Receive Overrun Interrupt Enable**

0: No effect.

1: Enables the Receive Overrun Interrupt.

- **CP0: Compare 0 Interrupt Enable**

0: No effect.

1: Enables the Compare 0 Interrupt.

- **CP1: Compare 1 Interrupt Enable**

0: No effect.

1: Enables the Compare 1 Interrupt.

- **TXSYN: Tx Sync Interrupt Enable**

0: No effect.

1: Enables the Tx Sync Interrupt.

Downloaded from Arrow.com.

- **RXSYN: Rx Sync Interrupt Enable**

0: No effect.

1: Enables the Rx Sync Interrupt.

Downloaded from Arrow.com.

43.9.15 SSC Interrupt Disable Register

Name: SSC_IDR

Address: 0x40004048

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
–	–	OVRUN	RXRDY	–	–	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready Interrupt Disable**

0: No effect.

1: Disables the Transmit Ready Interrupt.

- **TXEMPTY: Transmit Empty Interrupt Disable**

0: No effect.

1: Disables the Transmit Empty Interrupt.

- **RXRDY: Receive Ready Interrupt Disable**

0: No effect.

1: Disables the Receive Ready Interrupt.

- **OVRUN: Receive Overrun Interrupt Disable**

0: No effect.

1: Disables the Receive Overrun Interrupt.

- **CP0: Compare 0 Interrupt Disable**

0: No effect.

1: Disables the Compare 0 Interrupt.

- **CP1: Compare 1 Interrupt Disable**

0: No effect.

1: Disables the Compare 1 Interrupt.

- **TXSYN: Tx Sync Interrupt Enable**

0: No effect.

1: Disables the Tx Sync Interrupt.

Downloaded from Arrow.com.

- **RXSYN: Rx Sync Interrupt Enable**

0: No effect.

1: Disables the Rx Sync Interrupt.

Downloaded from Arrow.com.

43.9.16 SSC Interrupt Mask Register

Name: SSC_IMR

Address: 0x4000404C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
–	–	OVRUN	RXRDY	–	–	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready Interrupt Mask**

0: The Transmit Ready Interrupt is disabled.

1: The Transmit Ready Interrupt is enabled.

- **TXEMPTY: Transmit Empty Interrupt Mask**

0: The Transmit Empty Interrupt is disabled.

1: The Transmit Empty Interrupt is enabled.

- **RXRDY: Receive Ready Interrupt Mask**

0: The Receive Ready Interrupt is disabled.

1: The Receive Ready Interrupt is enabled.

- **OVRUN: Receive Overrun Interrupt Mask**

0: The Receive Overrun Interrupt is disabled.

1: The Receive Overrun Interrupt is enabled.

- **CP0: Compare 0 Interrupt Mask**

0: The Compare 0 Interrupt is disabled.

1: The Compare 0 Interrupt is enabled.

- **CP1: Compare 1 Interrupt Mask**

0: The Compare 1 Interrupt is disabled.

1: The Compare 1 Interrupt is enabled.

- **TXSYN: Tx Sync Interrupt Mask**

0: The Tx Sync Interrupt is disabled.

1: The Tx Sync Interrupt is enabled.

Downloaded from Arrow.com.

- **RXSYN: Rx Sync Interrupt Mask**

0: The Rx Sync Interrupt is disabled.

1: The Rx Sync Interrupt is enabled.

Downloaded from Arrow.com.

43.9.17 SSC Write Protection Mode Register

Name: SSC_WPMR

Address: 0x400040E4

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							

23	22	21	20	19	18	17	16
WPKEY							

15	14	13	12	11	10	9	8
WPKEY							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x535343 (“SSC” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x535343 (“SSC” in ASCII).

See Section 43.8.10 “Register Write Protection” for the list of registers that can be protected.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x535343	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

Downloaded from Arrow.com.

43.9.18 SSC Write Protection Status Register

Name: SSC_WPSR

Address: 0x400040E8

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

- **WPVS: Write Protection Violation Status**

0: No write protection violation has occurred since the last read of the SSC_WPSR.

1: A write protection violation has occurred since the last read of the SSC_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protect Violation Source**

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

Downloaded from Arrow.com.

44. Inter-IC Sound Controller (I2SC)

44.1 Description

The Inter-IC Sound Controller (I2SC) provides a 5-wire, bidirectional, synchronous, digital audio link to external audio devices: I2SDI, I2SDO, I2SWS, I2SCK, and I2SMCK pins.

The I2SC is compliant with the Inter-IC Sound (I^2S) bus specification.

The I2SC consists of a receiver, a transmitter and a common clock generator that can be enabled separately to provide Master, Slave or Controller modes with receiver and/or transmitter active.

DMA Controller channels, separate for the receiver and for the transmitter, allow a continuous high bit rate data transfer without processor intervention to the following:

- Audio CODECs in Master, Slave, or Controller mode
- Stereo DAC or ADC through a dedicated I^2S serial interface

The I2SC can use either a single DMA Controller channel for both audio channels or one DMA Controller channel per audio channel.

The 8- and 16-bit compact stereo format reduces the required DMA Controller bandwidth by transferring the left and right samples within the same data word.

In Master mode, the I2SC can produce a 32 f_s to 1024 f_s master clock that provides an over-sampling clock to an external audio codec or digital signal processor (DSP).

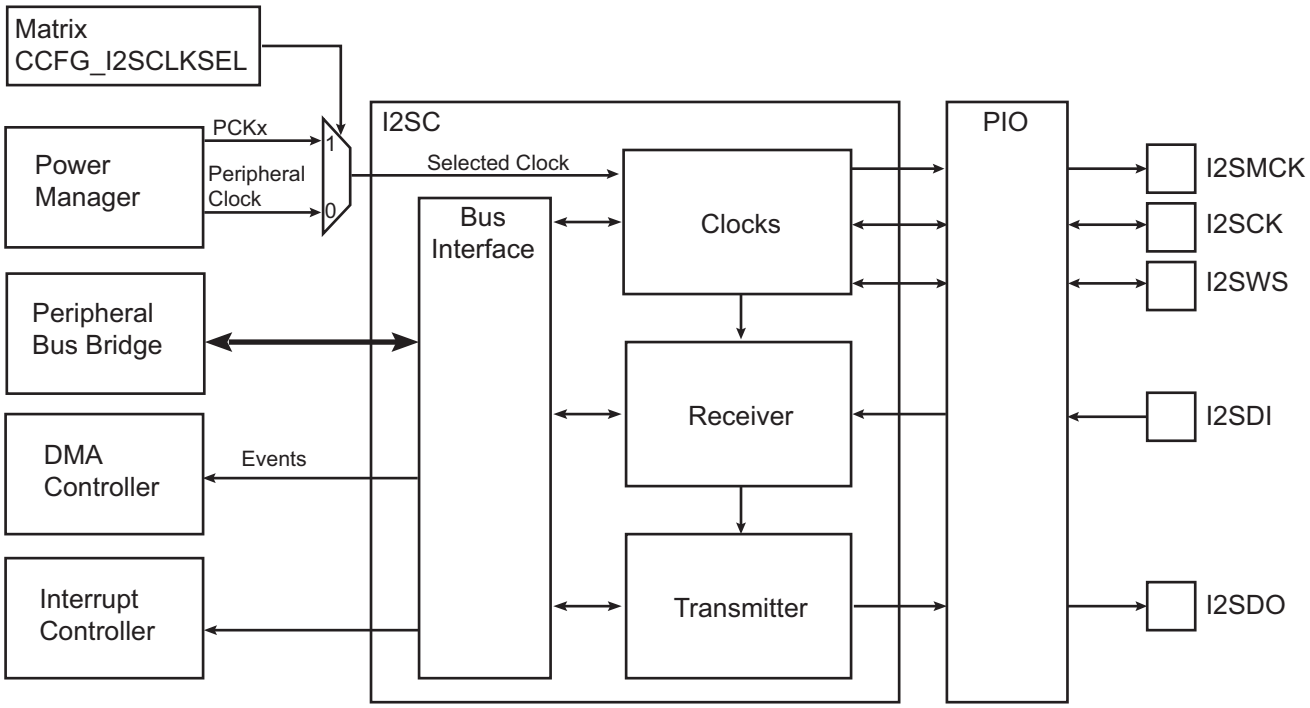
44.2 Embedded Characteristics

- Compliant with Inter-IC Sound (I^2S) Bus Specification
- Master, Slave, and Controller Modes
  - Slave: Data Received/Transmitted
  - Master: Data Received/Transmitted And Clocks Generated
  - Controller: Clocks Generated
- Individual Enable and Disable of Receiver, Transmitter and Clocks
- Configurable Clock Generator Common to Receiver and Transmitter
  - Suitable for a Wide Range of Sample Frequencies (f_s), Including 32 kHz, 44.1 kHz, 48 kHz, 88.2 kHz, 96 kHz, and 192 kHz
  - 32 f_s to 1024 f_s Master Clock Generated for External Oversampling Data Converters
- Support for Multiple Data Formats
  - 32-, 24-, 20-, 18-, 16-, and 8-bit Mono or Stereo Format
  - 16- and 8-bit Compact Stereo Format, with Left and Right Samples Packed in the Same Word to Reduce Data Transfers
- DMA Controller Interfaces the Receiver and Transmitter to Reduce Processor Overhead
  - One DMA Controller Channel for Both Audio Channels, or
  - One DMA Controller Channel Per Audio Channel
- Smart Holding Registers Management to Avoid Audio Channels Mix After Overrun or Underrun

Downloaded from Arrow.com.

44.3 Block Diagram

Figure 44-1. I2SC Block Diagram



44.4 I/O Lines Description

Table 44-1. I/O Lines Description

Pin Name	Pin Description	Type
I2SMCK	Master Clock	Output
I2SCK	Serial Clock	Input/Output
I2SWS	I^2S Word Select	Input/Output
I2SDI	Serial Data Input	Input
I2SDO	Serial Data Output	Output

Downloaded from Arrow.com.

44.5 Product Dependencies

To use the I2SC, other parts of the system must be configured correctly, as described below.

44.5.1 I/O Lines

The I2SC pins may be multiplexed with I/O Controller lines. The user must first program the PIO Controller to assign the required I2SC pins to their peripheral function. If the I2SC I/O lines are not used by the application, they can be used for other purposes by the PIO Controller. The user must enable the I2SC inputs and outputs that are used.

Table 44-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
I2SC0	I2SC0_CK	PA1	D
I2SC0	I2SC0_DI0	PA16	D
I2SC0	I2SC0_DO0	PA30	D
I2SC0	I2SC0_MCK	PA0	D
I2SC0	I2SC0_WS	PA15	D
I2SC1	I2SC1_CK	PA20	D
I2SC1	I2SC1_DI0	PE2	C
I2SC1	I2SC1_DO0	PE1	C
I2SC1	I2SC1_MCK	PA19	D
I2SC1	I2SC1_WS	PE0	C

44.5.2 Power Management

If the CPU enters a Sleep mode that disables clocks used by the I2SC, the I2SC stops functioning and resumes operation after the system wakes up from Sleep mode.

44.5.3 Clocks

The clock for the I2SC bus interface is generated by the Power Management Controller (PMC). I2SC must be disabled before disabling the clock to avoid freezing the I2SC in an undefined state.

44.5.4 DMA Controller

The I2SC interfaces to the DMA Controller. Using the I2SC DMA functionality requires the DMA Controller to be programmed first.

44.5.5 Interrupt Sources

The I2SC interrupt line is connected to the Interrupt Controller. Using the I2SC interrupt requires the Interrupt Controller to be programmed first.

Table 44-3. Peripheral IDs

Instance	ID
I2SC0	69
I2SC1	70

Downloaded from Arrow.com.

44.6 Functional Description

44.6.1 Initialization

The I2SC features a receiver, a transmitter and a clock generator for Master and Controller modes. Receiver and transmitter share the same serial clock and word select.

Before enabling the I2SC, the selected configuration must be written to the I2SC Mode Register (I2SC_MR) and to the I2S Clock Source Selection register (CCFG_I2SCLKSEL) described in the MATRIX section. If the I2SC_MR.IMCKMODE bit is set, the I2SC_MR.IMCKFS field must be configured as described in Section 44.6.5 ”Serial Clock and Word Select Generation”.

Once the I2SC_MR has been written, the I2SC clock generator, receiver, and transmitter can be enabled by writing a ’1’ to the CKEN, RXEN, and TXEN bits in the Control Register (I2SC_CR). The clock generator can be enabled alone in Controller mode to output clocks to the I2SMCK, I2SCK, and I2SWS pins. The clock generator must also be enabled if the receiver or the transmitter is enabled.

The clock generator, receiver, and transmitter can be disabled independently by writing a ’1’ to I2SC_CR.CXDIS, I2SC_CR.RXDIS and/or I2SC_CR.TXDIS, respectively. Once requested to stop, they stop only when the transmission of the pending frame transmission is completed.

44.6.2 Basic Operation

The receiver can be operated by reading the Receiver Holding Register (I2SC_RHR), whenever the Receive Ready (RXRDY) bit in the Status Register (I2SC_SR) is set. Successive values read from RHR correspond to the samples from the left and right audio channels for the successive frames.

The transmitter can be operated by writing to the Transmitter Holding Register (I2SC_THR), whenever the Transmit Ready (TXRDY) bit in the I2SC_SR is set. Successive values written to THR correspond to the samples from the left and right audio channels for the successive frames.

The RXRDY and TXRDY bits can be polled by reading the I2SC_SR.

The I2SC processor load can be reduced by enabling interrupt-driven operation. The RXRDY and/or TXRDY interrupt requests can be enabled by writing a ’1’ to the corresponding bit in the Interrupt Enable Register (I2SC_IER). The interrupt service routine associated to the I2SC interrupt request is executed whenever the Receive Ready or the Transmit Ready status bit is set.

44.6.3 Master, Controller and Slave Modes

In Master and Controller modes, the I2SC provides the master clock, the serial clock and the word select. I2SMCK, I2SCK, and I2SWS pins are outputs.

In Controller mode, the I2SC receiver and transmitter are disabled. Only the clocks are enabled and used by an external receiver and/or transmitter.

In Slave mode, the I2SC receives the serial clock and the word select from an external master. I2SCK and I2SWS pins are inputs.

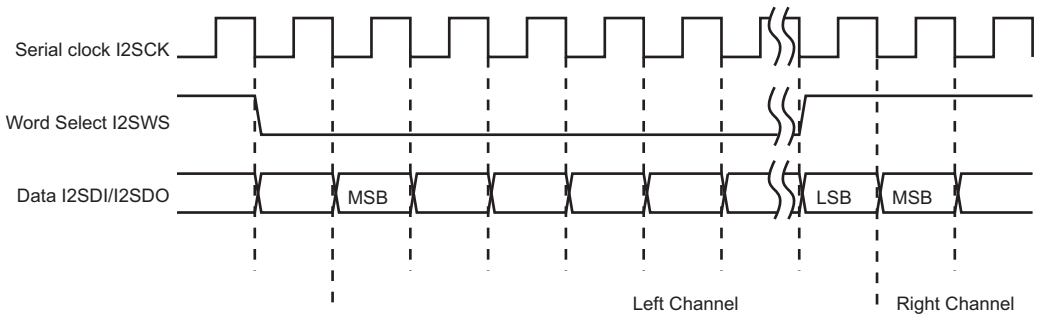
The mode is selected by writing the MODE field in the I2SC_MR. Since the MODE field changes the direction of the I2SWS and I2SSCK pins, the I2SC_MR must be written when the I2SC is stopped.

44.6.4 I^2S Reception and Transmission Sequence

As specified in the I^2S protocol, data bits are left-justified in the word select time slot, with the MSB transmitted first, starting one clock period after the transition on the word select line.

Downloaded from Arrow.com.

Figure 44-2. I²S Reception and Transmission Sequence



Data bits are sent on the falling edge of the serial clock and sampled on the rising edge of the serial clock. the word select line indicates the channel in transmission, a low level for the left channel and a high level for the right channel.

The length of transmitted words can be chosen among 8, 16, 18, 20, 24, and 32 bits by writing the I2SC_MR.DATALENGTH field.

If the time slot allows for more data bits than written in the I2SC_MR.DATALENGTH field, zeroes are appended to the transmitted data word or extra received bits are discarded.

44.6.5 Serial Clock and Word Select Generation

The generation of clocks in the I2SC is described in Figure 44-3 ”I2SC Clock Generation”.

In Slave mode, the serial clock and word select clock are driven by an external master. I2SCK and I2SWS pins are inputs.

In Master mode, the user can configure the master clock, serial clock, and word select clock through the I2SC_MR. I2SMCK, I2SCK, and I2SWS pins are outputs and MCK is used to derive the I2SC clocks.
In Master mode, if the Peripheral clock frequency is higher than 96 MHz, the PCKx clock from PMC must be selected as I2SC input clock by writing a ’1’ in the CLKSELx bit of the CCFG_I2CLKSEL register located in Matrix (See Figure 44-3 ”I2SC Clock Generation”).

Audio codecs connected to the I2SC pins may require a master clock (I2SMCK) signal with a frequency multiple of the audio sample frequency (f_s), such as $256f_s$. When the I2SC is in Master mode, writing a ’1’ to I2SC_MR.IMCKMODE outputs MCK as master clock to the I2SMCK pin, and divides MCK to create the internal bit clock, output on the I2SCK pin. The clock division factor is defined by writing to I2SC_MR.IMCKFS and I2SC_MR.DATALENGTH, as described in the I2SC_MR.IMCKFS field description.

The master clock (I2SMCK) frequency is [2×16 × (IMCKFS + 1)] / (IMCKDIV + 1) times the sample frequency (f_s), i.e., I2SWS frequency.

Example: If the sampling rate is 44.1 kHz with an I2S master clock (I2SMCK) ratio of 256, the core frequency must be an integer multiple of 11.2896 MHz. Assuming an integer multiple of 4, the IMCKDIV field must be configured to 4; the field IMCKFS must then be set to 31.

The serial clock (I2SCK) frequency is 2 × Slot Length times the sample frequency (f_s), where Slot Length is defined in Table 44-4.

Table 44-4. Slot Length

I2SC_MR.DATALENGTH	Word Length	Slot Length
0	32 bits	32

Downloaded from Arrow.com.

Table 44-4. Slot Length (Continued)

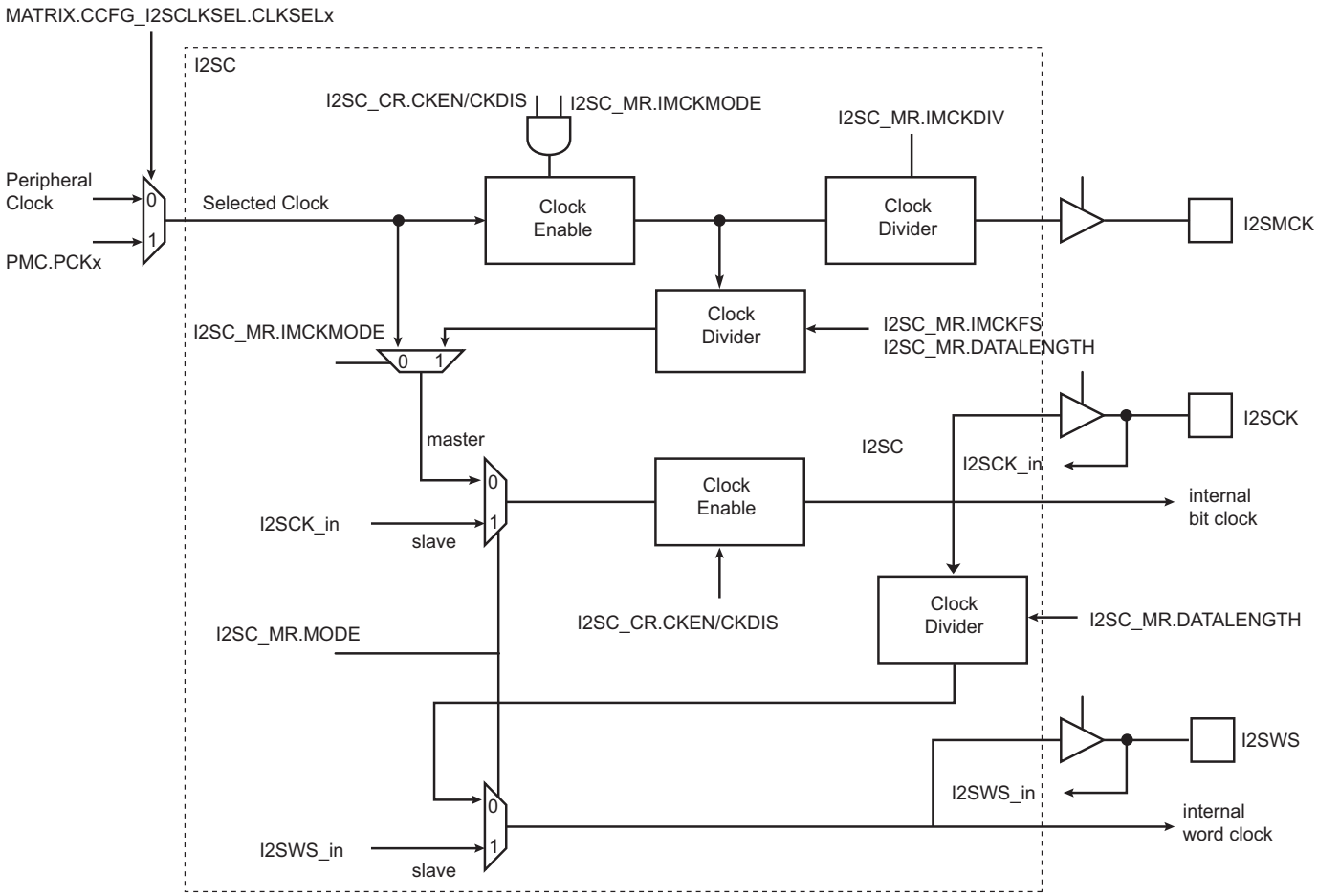
I2SC_MR.DATALENGTH	Word Length	Slot Length
1	24 bits	32 if I2SC_MR.IWS = 0 24 if I2SC_MR.IWS = 1
2	20 bits	
3	18 bits	
4	16 bits	16
5	16 bits compact stereo	
6	8 bits	8
7	8 bits compact stereo	

Warning: I2SC_MR.IMCKMODE must be written to '1' if the master clock frequency is strictly higher than the serial clock.

If a master clock output is not required, the MCK clock is used as I2SCK by clearing I2SC_MR.IMCKMODE. Alternatively, if the frequency of the MCK clock used is a multiple of the required I2SCK frequency, the I2SMCK to I2SCK divider can be used with the ratio defined by writing the I2SC_MR.IMCKFS field.

The I2SWS pin is used as word select as described in Section 44.6.4 "I2S Reception and Transmission Sequence".

Figure 44-3. I2SC Clock Generation



Downloaded from Arrow.com.

44.6.6 Mono

When the Transmit Mono bit (TXMONO) in I2SC_MR is set, data written to the left channel is duplicated to the right output channel.

When the Receive Mono bit (RXMONO) in I2SC_MR is set, data received from the left channel is duplicated to the right channel.

44.6.7 Holding Registers

The I2SC user interface includes a Receive Holding Register (I2SC_RHR) and a Transmit Holding Register (I2SC_THR). These registers are used to access audio samples for both audio channels.

When a new data word is available in I2SC_RHR, the Receive Ready bit (RXRDY) in I2SC_SR is set. Reading I2SC_RHR clears this bit.

A receive overrun condition occurs if a new data word becomes available before the previous data word has been read from I2SC_RHR. In this case, the Receive Overrun bit in I2SC_SR and bit i of the RXORCH field in I2SC_SR are set, where i is the current receive channel number.

When I2SC_THR is empty, the Transmit Ready bit (TXRDY) in I2SC_SR is set. Writing to I2SC_THR clears this bit.

A transmit underrun condition occurs if a new data word needs to be transmitted before it has been written to I2SC_THR. In this case, the Transmit Underrun (TXUR) bit and bit i of the TXORCH field in I2SC_SR are set, where i is the current transmit channel number. If the TXSAME bit in I2SC_MR is ’0’, then a zero data word is transmitted in case of underrun. If I2SC_MR.TXSAME is ’1’, then the previous data word for the current transmit channel number is transmitted.

Data words are right-justified in I2SC_RHR and I2SC_THR. For the 16-bit compact stereo data format, the left sample uses bits 15:0 and the right sample uses bits 31:16 of the same data word. For the 8-bit compact stereo data format, the left sample uses bits 7:0 and the right sample uses bits 15:8 of the same data word.

44.6.8 DMA Controller Operation

All receiver audio channels can be assigned to a single DMA Controller channel or individual audio channels can be assigned to one DMA Controller channel per audio channel. The same channel assignment choice applies to the transmitter audio channels.

Channel assignment is selected by writing to the I2SC_MR.RXDMA and I2SC_MR.TXDMA bits. If a single DMA Controller channel is selected, all data samples use I2SC receiver or transmitter DMA Controller channel 0.

The DMA Controller reads from the I2SC_RHR and writes to the I2SC_THR for both audio channels successively.

The DMA Controller transfers may use 32-bit word, 16-bit halfword, or 8-bit byte depending on the value of the I2SC_MR.DATALENGTH field.

44.6.9 Loop-back Mode

For debug purposes, the I2SC can be configured to loop back the transmitter to the Receiver. Writing a ’1’ to the I2SC_MR.LOOP bit internally connects I2SDO to I2SDI, so that the transmitted data is also received. Writing a ’0’ to I2SC_MR.LOOP restores the normal behavior with independent Receiver and Transmitter. As for other changes to the Receiver or Transmitter configuration, the I2SC Receiver and Transmitter must be disabled before writing to I2SC_MR to update I2SC_MR.LOOP.

44.6.10 Interrupts

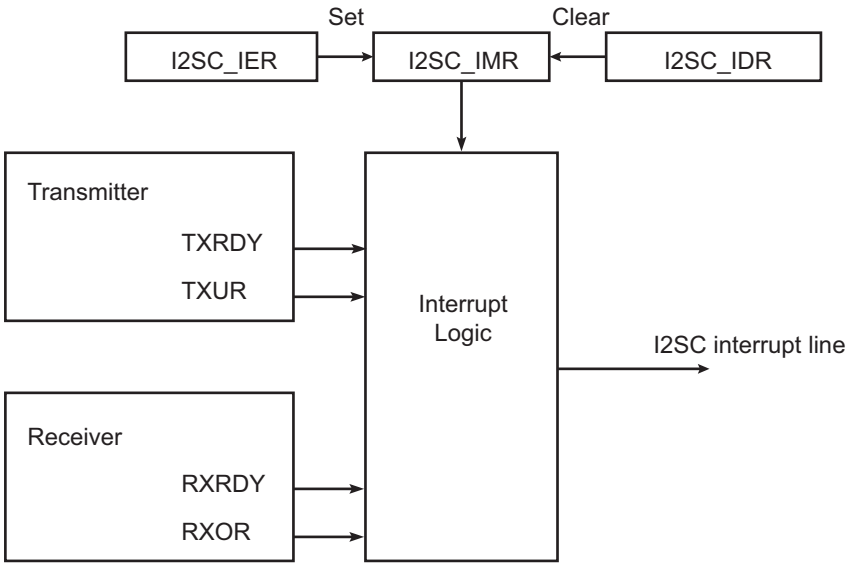
An I2SC interrupt request can be triggered whenever one or several of the following bits are set in I2SC_SR: Receive Ready (RXRDY), Receive Overrun (RXOR), Transmit Ready (TXRDY) or Transmit Underrun (TXUR).

Downloaded from Arrow.com.

The interrupt request is generated if the corresponding bit in the Interrupt Mask Register (I2SC_IMR) is set. Bits in I2SC_IMR are set by writing a ’1’ to the corresponding bit in I2SC_IER and cleared by writing a ’1’ to the corresponding bit in the Interrupt Disable Register (I2SC_IDR). The interrupt request remains active until the corresponding bit in I2SC_SR is cleared by writing a ’1’ to the corresponding bit in the Status Clear Register (I2SC_SCR).

For debug purposes, interrupt requests can be simulated by writing a ’1’ to the corresponding bit in the Status Set Register (I2SC_SSR).

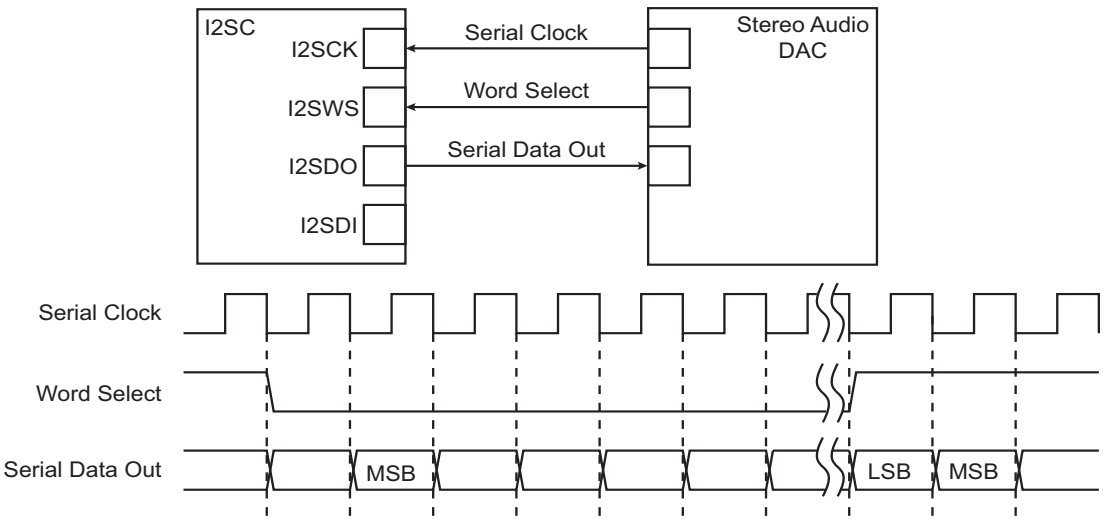
Figure 44-4. Interrupt Block Diagram



44.7 I2SC Application Examples

The I2SC supports several serial communication modes used in audio or high-speed serial links. Examples of standard applications are shown in the following figures. All serial link applications supported by the I2SC are not listed here.

Figure 44-5. Slave Transmitter I2SC Application Example



Downloaded from Arrow.com.

Figure 44-6. Dual Microphone Application Block Diagram

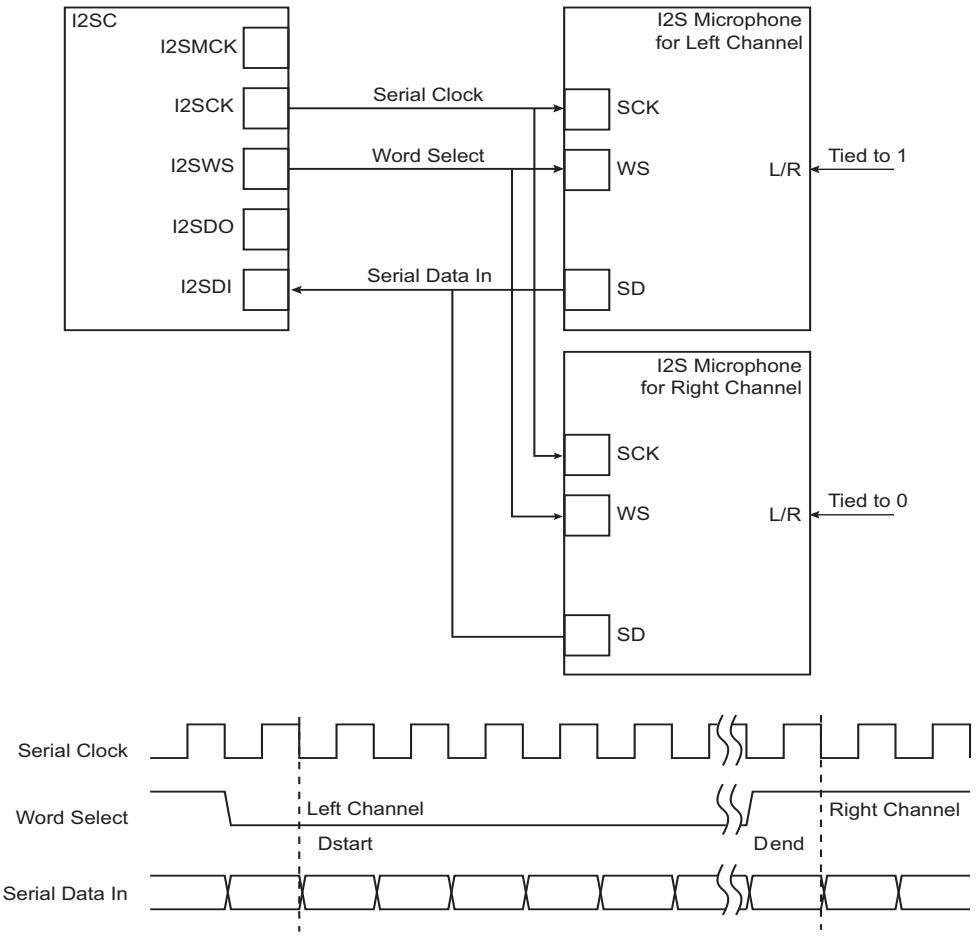
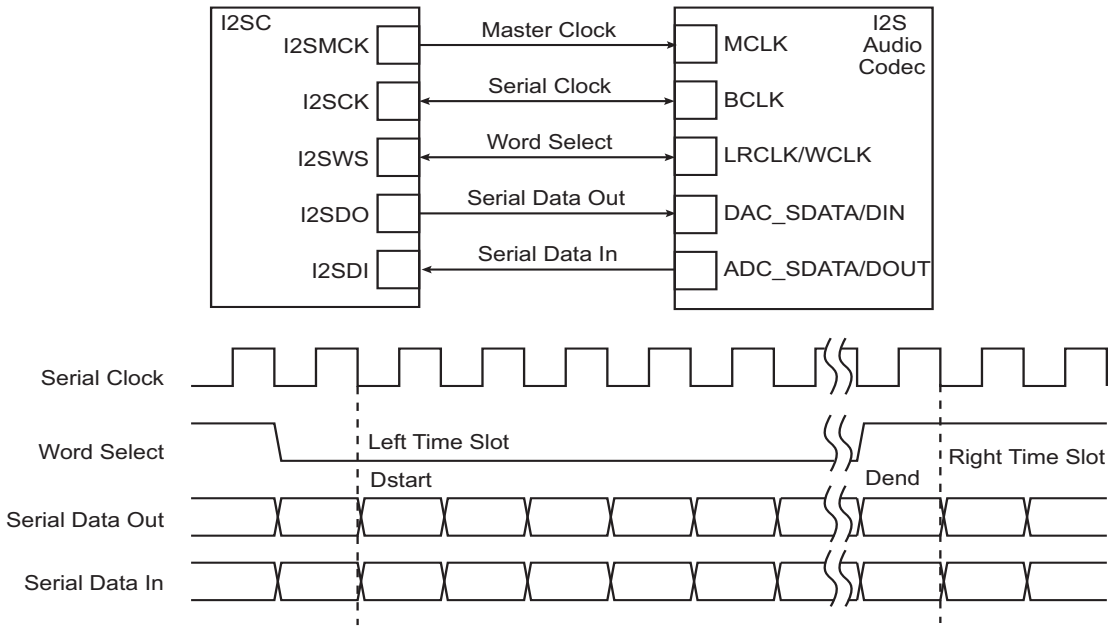


Figure 44-7. Codec Application Block Diagram



Downloaded from Arrow.com.

44.8 Inter-IC Sound Controller (I2SC) User Interface

Table 44-5. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	I2SC_CR	Write-only	–
0x04	Mode Register	I2SC_MR	Read/Write	0x00000000
0x08	Status Register	I2SC_SR	Read-only	0x00000000
0x0C	Status Clear Register	I2SC_SCR	Write-only	–
0x10	Status Set Register	I2SC_SSR	Write-only	–
0x14	Interrupt Enable Register	I2SC_IER	Write-only	–
0x18	Interrupt Disable Register	I2SC_IDR	Write-only	–
0x1C	Interrupt Mask Register	I2SC_IMR	Read-only	0x00000000
0x20	Receiver Holding Register	I2SC_RHR	Read-only	0x00000000
0x24	Transmitter Holding Register	I2SC_THR	Write-only	–
0x28–0xFC	Reserved	–	–	–

Downloaded from Arrow.com.

44.8.1 Inter-IC Sound Controller Control Register

Name: I2SC_CR

Address: 0x4008C000 (0), 0x40090000 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
SWRST	–	TXDIS	TXEN	CKDIS	CKEN	RXDIS	RXEN

- **RXEN: Receiver Enable**

0: Writing a ’0’ to this bit has no effect.

1: Writing a ’1’ to this bit enables the I2SC receiver, if RXDIS is not one. Bit I2SC_SR.RXEN is set when the receiver is activated.

- **RXDIS: Receiver Disable**

0: Writing a ’0’ to this bit has no effect.

1: Writing a ’1’ to this bit disables the I2SC receiver. Bit I2SC_SR.RXEN is cleared when the receiver is stopped.

- **CKEN: Clocks Enable**

0: Writing a ’0’ to this bit has no effect.

1: Writing a ’1’ to this bit enables the I2SC clocks generation, if CKDIS is not one.

- **CKDIS: Clocks Disable**

0: Writing a ’0’ to this bit has no effect.

1: Writing a zone to this bit disables the I2SC clock generation.

- **TXEN: Transmitter Enable**

0: Writing a ’0’ to this bit has no effect.

1: Writing a ’1’ to this bit enables the I2SC transmitter, if TXDIS is not one. Bit I2SC_SR.TXEN is set when the Transmitter is started.

- **TXDIS: Transmitter Disable**

0: Writing a ’0’ to this bit has no effect.

1: Writing a ’1’ to this bit disables the I2SC transmitter. Bit I2SC_SR.TXEN is cleared when the Transmitter is stopped.

- **SWRST: Software Reset**

0: Writing a ’0’ to this bit has no effect.

1: Writing a ’1’ to this bit resets all the registers in the I2SC. The I2SC is disabled after the reset.

Downloaded from Arrow.com.

44.8.2 Inter-IC Sound Controller Mode Register

Name: I2SC_MR

Address: 0x4008C004 (0), 0x40090004 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
IWS	IMCKMODE	IMCKFS					
23	22	21	20	19	18	17	16
–	–	IMCKDIV					
15	14	13	12	11	10	9	8
–	TXSAME	TXDMA	TXMONO	–	RXLOOP	RXDMA	RXMONO
7	6	5	4	3	2	1	0
FORMAT		–	DATALENGTH			–	MODE

The I2SC_MR must be written when the I2SC is stopped. The proper sequence is to write to I2SC_MR, then write to I2SC_CR to enable the I2SC or to disable the I2SC before writing a new value to I2SC_MR.

- **MODE: Inter-IC Sound Controller Mode**

Value	Name	Description
0	SLAVE	I2SCK and i2SWS pin inputs used as bit clock and word select/frame synchronization.
1	MASTER	Bit clock and word select/frame synchronization generated by I2SC from MCK and output to I2SCK and I2SWS pins. MCK is output as master clock on I2SMCK if I2SC_MR.IMCKMODE is set.

- **DATALENGTH: Data Word Length**

Value	Name	Description
0	32_BITS	Data length is set to 32 bits
1	24_BITS	Data length is set to 24 bits
2	20_BITS	Data length is set to 20 bits
3	18_BITS	Data length is set to 18 bits
4	16_BITS	Data length is set to 16 bits
5	16_BITS_COMPACT	Data length is set to 16-bit compact stereo. Left sample in bits 15:0 and right sample in bits 31:16 of same word.
6	8_BITS	Data length is set to 8 bits
7	8_BITS_COMPACT	Data length is set to 8-bit compact stereo. Left sample in bits 7:0 and right sample in bits 15:8 of the same word.

- **FORMAT: Data Format**

Value	Name	Description
0	I2S	I^2S format, stereo with I2SWS low for left channel, and MSB of sample starting one I2SCK period after I2SWS edge
1	LJ	Left-justified format, stereo with I2SWS high for left channel, and MSB of sample starting on I2SWS edge
2	–	Reserved
3	–	Reserved

Downloaded from Arrow.com.

- **RXDMA: Single or Multiple DMA Controller Channels for Receiver**

0: The receiver uses only one DMA Controller channel for all audio channels.

1: The receiver uses one DMA Controller channel per audio channel.

- **RXMONO: Receive Mono**

0: Stereo

1: Mono, with left audio samples duplicated to right audio channel by the I2SC.

- **RXLOOP: Loop-back Test Mode**

0: Normal mode

1: I2SDO output of I2SC is internally connected to I2SDI input.

- **TXMONO: Transmit Mono**

0: Stereo

1: Mono, with left audio samples duplicated to right audio channel by the I2SC.

- **TXDMA: Single or Multiple DMA Controller Channels for Transmitter**

0: The transmitter uses only one DMA Controller channel for all audio channels.

1: The transmitter uses one DMA Controller channel per audio channel.

- **TXSAME: Transmit Data when Underrun**

0: Zero sample transmitted when underrun.

1: Previous sample transmitted when underrun

- **IMCKDIV: Selected Clock to I2SC Master Clock Ratio**

I2SMCK Master clock output frequency is Selected Clock divided by (IMCKDIV + 1). Refer to the IMCKFS field description.

Notes: 1. This field is write-only. Always read as ‘0’.
2. Do not write a ‘0’ to this field.

- **IMCKFS: Master Clock to f_s Ratio**

Master clock frequency is [2 x 16 × (IMCKFS + 1)] / (IMCKDIV + 1) times the sample rate, i.e., I2SWS frequency.

Value	Name	Description
0	M2SF32	Sample frequency ratio set to 32
1	M2SF64	Sample frequency ratio set to 64
2	M2SF96	Sample frequency ratio set to 96
3	M2SF128	Sample frequency ratio set to 128
5	M2SF192	Sample frequency ratio set to 192
7	M2SF256	Sample frequency ratio set to 256
11	M2SF384	Sample frequency ratio set to 384
15	M2SF512	Sample frequency ratio set to 512
23	M2SF768	Sample frequency ratio set to 768
31	M2SF1024	Sample frequency ratio set to 1024
47	M2SF1536	Sample frequency ratio set to 1536
63	M2SF2048	Sample frequency ratio set to 2048

Downloaded from Arrow.com.

- **IMCKMODE: Master Clock Mode**

0: No master clock generated (Selected Clock drives I2SCK output).

1: Master clock generated (internally generated clock is used as I2SMCK output).

Warning: If I2SMCK frequency is the same as I2SCK, IMCKMODE must be cleared. Refer to Section 44.6.5 ”Serial Clock and Word Select Generation” and Table 44-4 “Slot Length”.

- **IWS: I2SWS Slot Width**

0: I2SWS slot is 32 bits wide for DATALENGTH = 18/20/24 bits.

1: I2SWS slot is 24 bits wide for DATALENGTH = 18/20/24 bits.

Refer to Table 44-4 “Slot Length”.

Downloaded from Arrow.com.

44.8.3 Inter-IC Sound Controller Status Register

Name: I2SC_SR

Address: 0x4008C008 (0), 0x40090008 (1)

Access: Read-only

31	30	29	28	27	26	25	24
	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	TXURCH			–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	RXORCH	
7	6	5	4	3	2	1	0
	TXUR	TXRDY	TXEN		RXOR	RXRDY	RXEN

- **RXEN: Receiver Enabled**

0: This bit is cleared when the receiver is disabled, following a RXDIS or SWRST request in I2SC_CR.

1: This bit is set when the receiver is enabled, following a RXEN request in I2SC_CR.

- **RXRDY: Receive Ready**

0: This bit is cleared when I2SC_RHR is read.

1: This bit is set when received data is present in I2SC_RHR.

- **RXOR: Receive Overrun**

0: This bit is cleared when the corresponding bit in I2SC_SCR is written to ’1’.

1: This bit is set when an overrun error occurs on I2SC_RHR or when the corresponding bit in I2SC_SSR is written to ’1’.

- **TXEN: Transmitter Enabled**

0: This bit is cleared when the transmitter is disabled, following a I2SC_CR.TXDIS or I2SC_CR.SWRST request.

1: This bit is set when the transmitter is enabled, following a I2SC_CR.TXEN request.

- **TXRDY: Transmit Ready**

0: This bit is cleared when data is written to I2SC_THR.

1: This bit is set when I2SC_THR is empty and can be written with new data to be transmitted.

- **TXUR: Transmit Underrun**

0: This bit is cleared when the corresponding bit in I2SC_SCR is written to ’1’.

1: This bit is set when an underrun error occurs on I2SC_THR or when the corresponding bit in I2SC_SSR is written to ’1’.

- **RXORCH: Receive Overrun Channel**

This field is cleared when I2SC_SCR.RXOR is written to ’1’.

Bit i of this field is set when a receive overrun error occurred in channel i (i = 0 for first channel of the frame).

- **TXURCH: Transmit Underrun Channel**

0: This field is cleared when I2SC_SCR.TXUR is written to ’1’.

1: Bit i of this field is set when a transmit underrun error occurred in channel i (i = 0 for first channel of the frame).

Downloaded from Arrow.com.

44.8.4 Inter-IC Sound Controller Status Clear Register

Name: I2SC_SCR

Address: 0x4008C00C (0), 0x4009000C (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	TXURCH		–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	RXORCH	
7	6	5	4	3	2	1	0
–	TXUR	–	–	–	RXOR	–	–

- **RXOR: Receive Overrun Status Clear**

Writing a ’0’ to this bit has no effect.

Writing a ’1’ to this bit clears the status bit.

- **TXUR: Transmit Underrun Status Clear**

Writing a ’0’ to this bit has no effect.

Writing a ’1’ to this bit clears the status bit.

- **RXORCH: Receive Overrun Per Channel Status Clear**

Writing a ’0’ has no effect.

Writing a ’1’ to any bit in this field clears the corresponding bit in the I2SC_SR and the corresponding interrupt request.

- **TXURCH: Transmit Underrun Per Channel Status Clear**

Writing a ’0’ has no effect.

Writing a ’1’ to any bit in this field clears the corresponding bit in the I2SC_SR and the corresponding interrupt request.

Downloaded from Arrow.com.

44.8.5 Inter-IC Sound Controller Status Set Register

Name: I2SC_SSR

Address: 0x4008C010 (0), 0x40090010 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	TXURCH		–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	RXORCH	
7	6	5	4	3	2	1	0
–	TXUR	–	–	–	RXOR	–	–

- **RXOR: Receive Overrun Status Set**

Writing a ’0’ to this bit has no effect.

Writing a ’1’ to this bit sets the status bit.

- **TXUR: Transmit Underrun Status Set**

Writing a ’0’ to this bit has no effect.

Writing a ’1’ to this bit sets the status bit.

- **RXORCH: Receive Overrun Per Channel Status Set**

Writing a ’0’ has no effect.

Writing a ’1’ to any bit in this field sets the corresponding bit in I2SC_SR and the corresponding interrupt request.

- **TXURCH: Transmit Underrun Per Channel Status Set**

Writing a ’0’ has no effect.

Writing a ’1’ to any bit in this field sets the corresponding bit in I2SC_SR and the corresponding interrupt request.

Downloaded from Arrow.com.

44.8.6 Inter-IC Sound Controller Interrupt Enable Register

Name: I2SC_IER

Address: 0x4008C014 (0), 0x40090014 (1)

Access: Write-only

31	30	29	28	27	26	25	24
	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–		–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
	TXUR	TXRDY	–		RXOR	RXRDY	–

- **RXRDY: Receiver Ready Interrupt Enable**

0: Writing a ’0’ to this bit has no effect.

1: Writing a ’1’ to this bit sets the corresponding bit in I2SC_IMR.

- **RXOR: Receiver Overrun Interrupt Enable**

0: Writing a ’0’ to this bit has no effect.

1: Writing a ’1’ to this bit sets the corresponding bit in I2SC_IMR.

- **TXRDY: Transmit Ready Interrupt Enable**

0: Writing a ’0’ to this bit has no effect.

1: Writing a ’1’ to this bit sets the corresponding bit in I2SC_IMR.

- **TXUR: Transmit Underflow Interrupt Enable**

0: Writing a ’0’ to this bit has no effect.

1: Writing a ’1’ to this bit sets the corresponding bit in I2SC_IMR.

Downloaded from Arrow.com.

44.8.7 Inter-IC Sound Controller Interrupt Disable Register

Name: I2SC_IDR

Address: 0x4008C018 (0), 0x40090018 (1)

Access: Write-only

31	30	29	28	27	26	25	24
	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–		–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
	TXUR	TXRDY	–		RXOR	RXRDY	–

- **RXRDY: Receiver Ready Interrupt Disable**

0: Writing a ’0’ to this bit has no effect.

1: Writing a ’1’ to this bit clears the corresponding bit in I2SC_IMR.

- **RXOR: Receiver Overrun Interrupt Disable**

0: Writing a ’0’ to this bit has no effect.

1: Writing a ’1’ to this bit clears the corresponding bit in I2SC_IMR.

- **TXRDY: Transmit Ready Interrupt Disable**

0: Writing a ’0’ to this bit has no effect.

1: Writing a ’1’ to this bit clears the corresponding bit in I2SC_IMR.

- **TXUR: Transmit Underflow Interrupt Disable**

0: Writing a ’0’ to this bit has no effect.

1: Writing a ’1’ to this bit clears the corresponding bit in I2SC_IMR.

Downloaded from Arrow.com.

44.8.8 Inter-IC Sound Controller Interrupt Mask Register

Name: I2SC_IMR

Address: 0x4008C01C (0), 0x4009001C (1)

Access: Write-only

31	30	29	28	27	26	25	24
	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–		–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
	TXUR	TXRDY	–		RXOR	RXRDY	–

- **RXRDY: Receiver Ready Interrupt Disable**

0: The corresponding interrupt is disabled. This bit is cleared when the corresponding bit in I2SC_IDR is written to ’1’.

1: The corresponding interrupt is enabled. This bit is set when the corresponding bit in I2SC_IER is written to ’1’.

- **RXOR: Receiver Overrun Interrupt Disable**

0: The corresponding interrupt is disabled. This bit is cleared when the corresponding bit in I2SC_IDR is written to ’1’.

1: The corresponding interrupt is enabled. This bit is set when the corresponding bit in I2SC_IER is written to ’1’.

- **TXRDY: Transmit Ready Interrupt Disable**

0: The corresponding interrupt is disabled. This bit is cleared when the corresponding bit in I2SC_IDR is written to ’1’.

1: The corresponding interrupt is enabled. This bit is set when the corresponding bit in I2SC_IER is written to ’1’.

- **TXUR: Transmit Underflow Interrupt Disable**

0: The corresponding interrupt is disabled. This bit is cleared when the corresponding bit in I2SC_IDR is written to ’1’.

1: The corresponding interrupt is enabled. This bit is set when the corresponding bit in I2SC_IER is written to ’1’.

Downloaded from Arrow.com.

44.8.9 Inter-IC Sound Controller Receiver Holding Register

Name: I2SC_RHR

Address: 0x4008C020 (0), 0x40090020 (1)

Access: Read-only

31	30	29	28	27	26	25	24
RHR							
23	22	21	20	19	18	17	16
RHR							
15	14	13	12	11	10	9	8
RHR							
7	6	5	4	3	2	1	0
RHR							

- **RHR: Receiver Holding Register**

This field is set by hardware to the last received data word. If I2SC_MR.DATALENGTH specifies fewer than 32 bits, data is right justified in the RHR field.

Downloaded from Arrow.com.

44.8.10 Inter-IC Sound Controller Transmitter Holding Register

Name: I2SC_THR

Address: 0x4008C024 (0), 0x40090024 (1)

Access: Write-only

31	30	29	28	27	26	25	24
THR							
23	22	21	20	19	18	17	16
THR							
15	14	13	12	11	10	9	8
THR							
7	6	5	4	3	2	1	0
THR							

- **THR: Transmitter Holding Register**

Next data word to be transmitted after the current word if TXRDY is not set. If I2SC_MR.DATALENGTH specifies fewer than 32 bits, data is right-justified in the THR field.

Downloaded from Arrow.com.

45. Universal Synchronous Asynchronous Receiver Transceiver (USART)

45.1 Description

The Universal Synchronous Asynchronous Receiver Transceiver (USART) provides one full duplex universal synchronous asynchronous serial link. Data frame format is widely programmable (data length, parity, number of stop bits) to support a maximum of standards. The receiver implements parity error, framing error and overrun error detection. The receiver time-out enables handling variable-length frames and the transmitter timeguard facilitates communications with slow remote devices. Multidrop communications are also supported through address bit handling in reception and transmission.

The USART features three test modes: Remote Loopback, Local Loopback and Automatic Echo.

The USART supports specific operating modes providing interfaces on RS485, LIN, LON, and SPI buses, with ISO7816 T = 0 or T = 1 smart card slots, infrared transceivers and connection to modem ports. The hardware handshaking feature enables an out-of-band flow control by automatic management of the pins RTS and CTS.

The USART supports the connection to the DMA Controller, which enables data transfers to the transmitter and from the receiver. The DMAC provides chained buffer management without any intervention of the processor.

45.2 Embedded Characteristics

- Programmable Baud Rate Generator
- 5- to 9-bit Full-duplex Synchronous or Asynchronous Serial Communications
  - 1, 1.5 or 2 Stop Bits in Asynchronous Mode or 1 or 2 Stop Bits in Synchronous Mode
  - Parity Generation and Error Detection
  - Framing Error Detection, Overrun Error Detection
  - Digital Filter on Receive Line
  - MSB- or LSB-first
  - Optional Break Generation and Detection
  - By 8 or by 16 Oversampling Receiver Frequency
  - Optional Hardware Handshaking RTS-CTS
  - Optional Modem Signal Management DTR-DSR-DCD-RI
  - Receiver Time-out and Transmitter Timeguard
  - Optional Multidrop Mode with Address Generation and Detection
- RS485 with Driver Control Signal
- ISO7816, T = 0 or T = 1 Protocols for Interfacing with Smart Cards
  - NACK Handling, Error Counter with Repetition and Iteration Limit
- IrDA Modulation and Demodulation
  - Communication at up to 115.2 kbit/s
- SPI Mode
  - Master or Slave
  - Serial Clock Programmable Phase and Polarity
  - SPI Serial Clock (SCK) Frequency up to $f_{peripheral\ clock}/6$
- LIN Mode
  - Compliant with LIN 1.3 and LIN 2.0 SPECIFICATIONS
  - Master or Slave
  - Processing of Frames with Up to 256 Data Bytes
  - Response Data Length can be Configurable or Defined Automatically by the Identifier

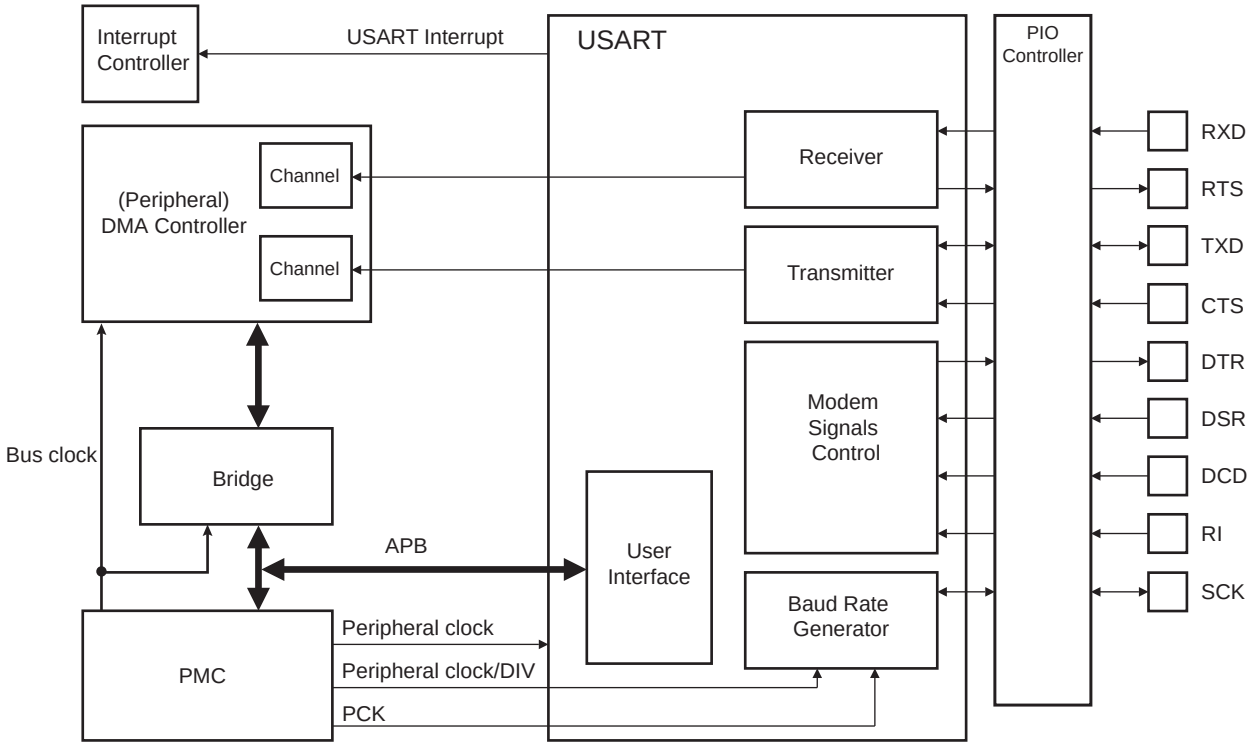
Downloaded from Arrow.com.

  - Self-synchronization in Slave Node Configuration
  - Automatic Processing and Verification of the “Synch Break” and the “Synch Field”
  - “Synch Break” Detection Even When Partially Superimposed with a Data Byte
  - Automatic Identifier Parity Calculation/Sending and Verification
  - Parity Sending and Verification Can be Disabled
  - Automatic Checksum Calculation/sending and Verification
  - Checksum Sending and Verification Can be Disabled
  - Support Both “Classic” and “Enhanced” Checksum Types
  - Full LIN Error Checking and Reporting
  - Frame Slot Mode: Master Allocates Slots to the Scheduled Frames Automatically
  - Generation of the Wakeup Signal
- LON Mode
  - Compliant with CEA-709 Specification
  - Full-layer 2 Implementation
  - Differential Manchester Encoding/Decoding (CDP)
  - Preamble Generation Including Bit- and Byte-sync Fields
  - LON Timings Handling (beta1, beta2, IDT, etc.)
  - CRC Generation and Checking
  - Automated Random Number Generation
  - Backlog Calculation and Update
  - Collision Detection Support
  - Supports Both *comm_type=1* and *comm_type=2* Modes
  - Clock Drift Tolerance Up to 16%
- Test Modes
  - Remote Loopback, Local Loopback, Automatic Echo
- Supports Connection of:
  - Two DMA Controller Channels (DMAC)
- Offers Buffer Transfer without Processor Intervention
- Register Write Protection

Downloaded from Arrow.com.

45.3 Block Diagram

Figure 45-1. USART Block Diagram



45.4 I/O Lines Description

Table 45-1. I/O Line Description

Name	Description	Type	Active Level
SCK	Serial Clock	I/O	—
TXD	Transmit Serial Data or Master Out Slave In (MOSI) in SPI Master mode or Master In Slave Out (MISO) in SPI Slave mode	I/O	—
RXD	Receive Serial Data or Master In Slave Out (MISO) in SPI Master mode or Master Out Slave In (MOSI) in SPI Slave mode	Input	—
RI	Ring Indicator	Input	Low
DSR	Data Set Ready	Input	Low
DCD	Data Carrier Detect	Input	Low
DTR	Data Terminal Ready	Output	Low
LCOL	LON Collision Detection	Input	Low
CTS	Clear to Send or Slave Select (NSS) in SPI Slave mode	Input	Low

Downloaded from Arrow.com.

Table 45-1. I/O Line Description (Continued)

Name	Description	Type	Active Level
RTS	Request to Send or Slave Select (NSS) in SPI Master mode	Output	Low

45.5 Product Dependencies

45.5.1 I/O Lines

The pins used for interfacing the USART may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the desired USART pins to their peripheral function. If I/O lines of the USART are not used by the application, they can be used for other purposes by the PIO Controller.

All the pins of the modems may or may not be implemented on the USART. Only USART3 is fully equipped with all the modem signals. On USARTs not equipped with the corresponding pin, the associated control bits and statuses have no effect on the behavior of the USART.

Table 45-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
USART0	CTS0	PB2	C
USART0	DCD0	PD0	D
USART0	DSR0	PD2	D
USART0	DTR0	PD1	D
USART0	RI0	PD3	D
USART0	RTS0	PB3	C
USART0	RXD0	PB0	C
USART0	SCK0	PB13	C
USART0	TXD0	PB1	C
USART1	CTS1	PA25	A
USART1	DCD1	PA26	A
USART1	DSR1	PA28	A
USART1	DTR1	PA27	A
USART1	LONCOL1	PA3	B
USART1	RI1	PA29	A
USART1	RTS1	PA24	A
USART1	RXD1	PA21	A
USART1	SCK1	PA23	A
USART1	TXD1	PB4	D
USART2	CTS2	PD19	B
USART2	DCD2	PD4	D
USART2	DSR2	PD6	D
USART2	DTR2	PD5	D
USART2	RI2	PD7	D

Downloaded from Arrow.com.

Table 45-2. I/O Lines

USART2	RTS2	PD18	B
USART2	RXD2	PD15	B
USART2	SCK2	PD17	B
USART2	TXD2	PD16	B

45.5.2 Power Management

The USART is not continuously clocked. The programmer must first enable the USART clock in the Power Management Controller (PMC) before using the USART. However, if the application does not require USART operations, the USART clock can be stopped when not needed and be restarted later. In this case, the USART will resume its operations where it left off.

45.5.3 Interrupt Sources

The USART interrupt line is connected on one of the internal sources of the Interrupt Controller. Using the USART interrupt requires the Interrupt Controller to be programmed first.

Table 45-3. Peripheral IDs

Instance	ID
USART0	13
USART1	14
USART2	15

45.6 Functional Description

45.6.1 Baud Rate Generator

The baud rate generator provides the bit period clock, also named the baud rate clock, to both the receiver and the transmitter.

The baud rate generator clock source is selected by configuring the USCLKS field in the USART Mode Register (US_MR) to one of the following:

- The peripheral clock
- A division of the peripheral clock, where the divider is product-dependent, but generally set to 8
- A processor/peripheral independent clock source fully programmable provided by PMC (PCK)
- The external clock, available on the SCK pin

The baud rate generator is based upon a 16-bit divider, which is programmed with the CD field of the Baud Rate Generator register (US_BRGR). If a 0 is written to CD, the baud rate generator does not generate any clock. If a 1 is written to CD, the divider is bypassed and becomes inactive.

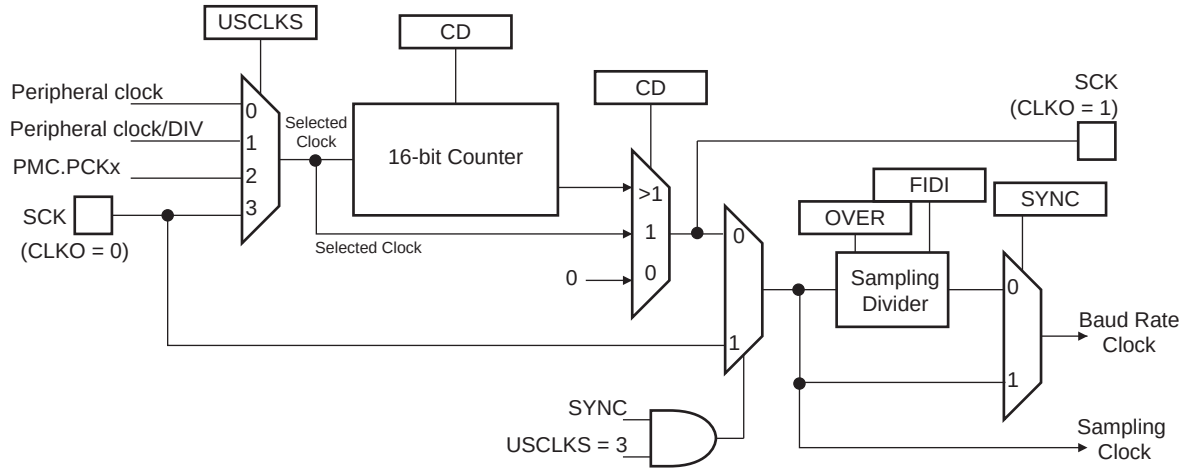
If the external SCK clock is selected, the duration of the low and high levels of the signal provided on the SCK pin must be longer than a peripheral clock period. The frequency of the signal provided on SCK must be at least 3 times lower than the frequency provided on the peripheral clock in USART mode (field USART_MODE differs from 0xE or 0xF), or 6 times lower in SPI mode (field USART_MODE equals 0xE or 0xF).

If PMC PCK is selected, the baud rate is independent of the processor/peripheral clock and thus processor/peripheral clock frequency can be changed without affecting the USART transfer. The PMC PCKx frequency must always be three times lower than the peripheral clock frequency.

If PMC PCK is selected (USCLKS = 2) and the SCK pin is driven (CLKO = 1), the CD field must be greater than 1.

Downloaded from Arrow.com.

Figure 45-2. Baud Rate Generator



45.6.1.1 Baud Rate in Asynchronous Mode

If the USART is programmed to operate in Asynchronous mode, the selected clock is first divided by CD, which is field programmed in the US_BRGR. The resulting clock is provided to the receiver as a sampling clock and then divided by 16 or 8, depending on how the OVER bit in the US_MR is programmed.

If OVER is set, the receiver sampling is eight times higher than the baud rate clock. If OVER is cleared, the sampling is performed at 16 times the baud rate clock.

The baud rate is calculated as per the following formula:

$$\text{Baud Rate} = \frac{\text{Selected Clock}}{(8(2-OVER)CD)}$$

This gives a maximum baud rate of peripheral clock divided by 8, assuming that the peripheral clock is the highest possible clock and that the OVER bit is set.

Baud Rate Calculation Example

Table 45-4 shows calculations of CD to obtain a baud rate at 38,400 bit/s for different source clock frequencies. This table also shows the actual resulting baud rate and the error.

Table 45-4. Baud Rate Example (OVER = 0)

Source Clock (MHz)	Expected Baud Rate (bit/s)	Calculation Result	CD	Actual Baud Rate (bit/s)	Error
3,686,400	38,400	6.00	6	38,400.00	0.00%
4,915,200	38,400	8.00	8	38,400.00	0.00%
5,000,000	38,400	8.14	8	39,062.50	1.70%
7,372,800	38,400	12.00	12	38,400.00	0.00%
8,000,000	38,400	13.02	13	38,461.54	0.16%
12,000,000	38,400	19.53	20	37,500.00	2.40%
12,288,000	38,400	20.00	20	38,400.00	0.00%
14,318,180	38,400	23.30	23	38,908.10	1.31%
14,745,600	38,400	24.00	24	38,400.00	0.00%
18,432,000	38,400	30.00	30	38,400.00	0.00%
24,000,000	38,400	39.06	39	38,461.54	0.16%

Downloaded from Arrow.com.

Table 45-4. Baud Rate Example (OVER = 0) (Continued)

Source Clock (MHz)	Expected Baud Rate (bit/s)	Calculation Result	CD	Actual Baud Rate (bit/s)	Error
24,576,000	38,400	40.00	40	38,400.00	0.00%
25,000,000	38,400	40.69	40	38,109.76	0.76%
32,000,000	38,400	52.08	52	38,461.54	0.16%
32,768,000	38,400	53.33	53	38,641.51	0.63%
33,000,000	38,400	53.71	54	38,194.44	0.54%
40,000,000	38,400	65.10	65	38,461.54	0.16%
50,000,000	38,400	81.38	81	38,580.25	0.47%
60,000,000	38,400	97.66	98	38,265.31	0.35%
70,000,000	38,400	113.93	114	38,377.19	0.06%

In this example, the baud rate is calculated with the following formula:

$$\text{Baud Rate} = \text{Selected Clock} / CD \times 16$$

The baud rate error is calculated with the following formula. It is not recommended to work with an error higher than 5%.

$$Error = 1 - \left(\frac{\text{Expected Baud Rate}}{\text{Actual Baud Rate}}\right)$$

45.6.1.2 Fractional Baud Rate in Asynchronous Mode

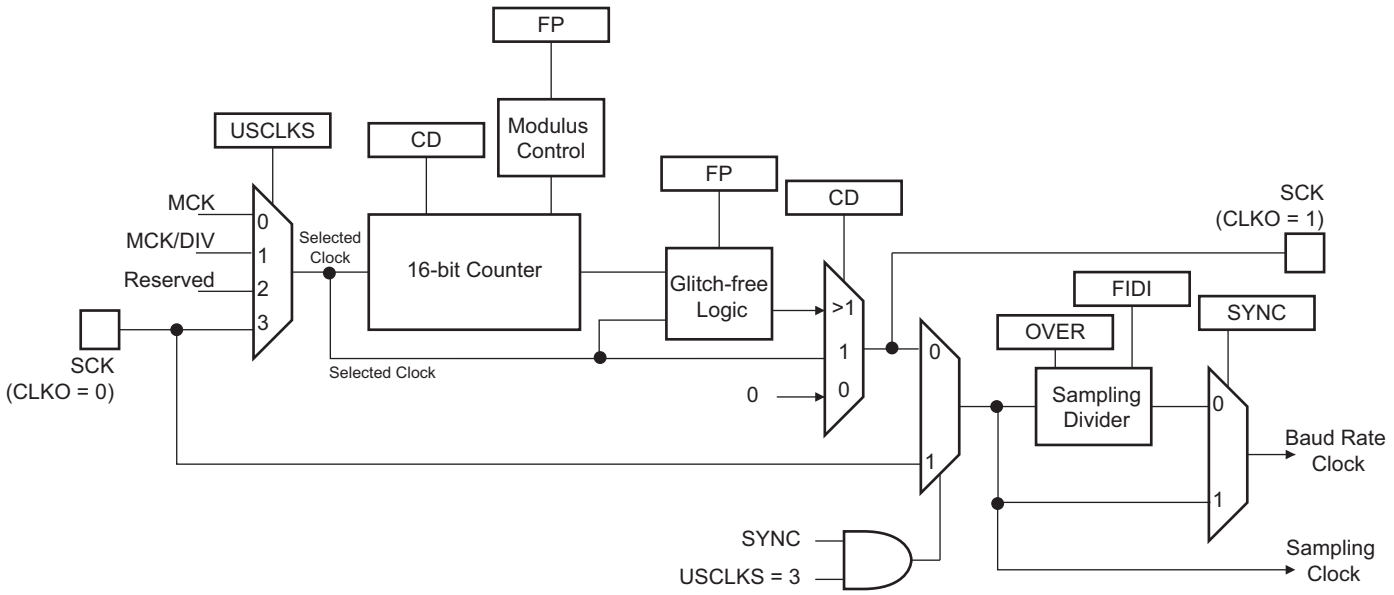
The baud rate generator is subject to the following limitation: the output frequency changes only by integer multiples of the reference frequency. An approach to this problem is to integrate a fractional N clock generator that has a high resolution. The generator architecture is modified to obtain baud rate changes by a fraction of the reference source clock. This fractional part is programmed with the FP field in the US_BRGR. If FP is not 0, the fractional part is activated. The resolution is one eighth of the clock divider. This feature is only available when using USART normal mode. The fractional baud rate is calculated using the following formula:

$$\text{Baud Rate} = \frac{\text{Selected Clock}}{\left(8(2 - OVER)\left(CD + \frac{FP}{8}\right)\right)}$$

The modified architecture is presented in the following Figure 45-3.

Downloaded from Arrow.com.

Figure 45-3. Fractional Baud Rate Generator



Warning: When the value of field FP is greater than 0, the SCK (oversampling clock) generates non-constant duty cycles. The SCK high duration is increased by “selected clock” period from time to time. The duty cycle depends on the value of the CD field.

45.6.1.3 Baud Rate in Synchronous Mode or SPI Mode

If the USART is programmed to operate in Synchronous mode, the selected clock is simply divided by the field CD in the US_BRGR.

$$\text{Baud Rate} = \frac{\text{Selected Clock}}{CD}$$

In Synchronous mode, if the external clock is selected (USCLKS = 3), the clock is provided directly by the signal on the USART SCK pin. No division is active. The value written in US_BRGR has no effect. The external clock frequency must be at least 3 times lower than the system clock. In Synchronous mode master (USCLKS = 0 or 1, CLKO set to 1), the receive part limits the SCK maximum frequency to Selected Clock/3 in USART mode, or Selected Clock/6 in SPI mode.

When either the external clock SCK or the internal clock divided (peripheral clock/DIV) is selected, the value programmed in CD must be even if the user has to ensure a 50:50 mark/space ratio on the SCK pin. When the peripheral clock is selected, the baud rate generator ensures a 50:50 duty cycle on the SCK pin, even if the value programmed in CD is odd.

45.6.1.4 Baud Rate in ISO 7816 Mode

The ISO7816 specification defines the bit rate with the following formula:

$$B = \frac{Di}{Fi} \times f$$

where:

- B is the bit rate
- Di is the bit-rate adjustment factor
- Fi is the clock frequency division factor
- f is the ISO7816 clock frequency (Hz)

Di is a binary value encoded on a 4-bit field, named DI, as represented in Table 45-5.

Downloaded from Arrow.com.

Table 45-5. Binary and Decimal Values for Di

DI field	0001	0010	0011	0100	0101	0110	1000	1001
Di (decimal)	1	2	4	8	16	32	12	20

Fi is a binary value encoded on a 4-bit field, named FI, as represented in Table 45-6.

Table 45-6. Binary and Decimal Values for Fi

FI field	0000	0001	0010	0011	0100	0101	0110	1001	1010	1011	1100	1101
Fi (decimal)	372	372	558	744	1116	1488	1860	512	768	1024	1536	2048

Table 45-7 shows the resulting Fi/Di ratio, which is the ratio between the ISO7816 clock and the baud rate clock.

Table 45-7. Possible Values for the Fi/Di Ratio

Fi/Di	372	558	744	1116	1488	1806	512	768	1024	1536	2048
1	372	558	744	1116	1488	1860	512	768	1024	1536	2048
2	186	279	372	558	744	930	256	384	512	768	1024
4	93	139.5	186	279	372	465	128	192	256	384	512
8	46.5	69.75	93	139.5	186	232.5	64	96	128	192	256
16	23.25	34.87	46.5	69.75	93	116.2	32	48	64	96	128
32	11.62	17.43	23.25	34.87	46.5	58.13	16	24	32	48	64
12	31	46.5	62	93	124	155	42.66	64	85.33	128	170.6
20	18.6	27.9	37.2	55.8	74.4	93	25.6	38.4	51.2	76.8	102.4

If the USART is configured in ISO7816 mode, the clock selected by the USCLKS field in US_MR is first divided by the value programmed in the field CD in the US_BRGR. The resulting clock can be provided to the SCK pin to feed the smart card clock inputs. This means that the CLKO bit can be set in US_MR.

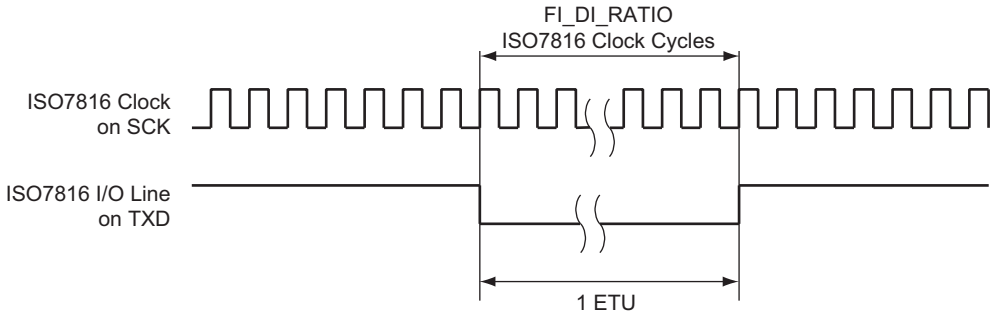
This clock is then divided by the value programmed in the FI_DI_RATIO field in the FI_DI_Ratio register (US_FIDI). This is performed by the Sampling Divider, which performs a division by up to 2047 in ISO7816 mode. The non-integer values of the Fi/Di Ratio are not supported and the user must program the FI_DI_RATIO field to a value as close as possible to the expected value.

The FI_DI_RATIO field resets to the value 0x174 (372 in decimal) and is the most common divider between the ISO7816 clock and the bit rate (Fi = 372, Di = 1).

Figure 45-4 shows the relation between the Elementary Time Unit, corresponding to a bit time, and the ISO 7816 clock.

Downloaded from Arrow.com.

Figure 45-4. Elementary Time Unit (ETU)



45.6.2 Receiver and Transmitter Control

After reset, the receiver is disabled. The user must enable the receiver by setting the RXEN bit in the Control register (US_CR). However, the receiver registers can be programmed before the receiver clock is enabled.

After reset, the transmitter is disabled. The user must enable it by setting the TXEN bit in the US_CR. However, the transmitter registers can be programmed before being enabled.

The receiver and the transmitter can be enabled together or independently.

At any time, the software can perform a reset on the receiver or the transmitter of the USART by setting the corresponding bit, RSTRX and RSTTX respectively, in the US_CR. The software resets clear the status flag and reset internal state machines but the user interface configuration registers hold the value configured prior to software reset. Regardless of what the receiver or the transmitter is performing, the communication is immediately stopped.

The user can also independently disable the receiver or the transmitter by setting RXDIS and TXDIS respectively in the US_CR. If the receiver is disabled during a character reception, the USART waits until the end of reception of the current character, then the reception is stopped. If the transmitter is disabled while it is operating, the USART waits the end of transmission of both the current character and character being stored in the Transmit Holding register (US_THR). If a timeguard is programmed, it is handled normally.

45.6.3 Synchronous and Asynchronous Modes

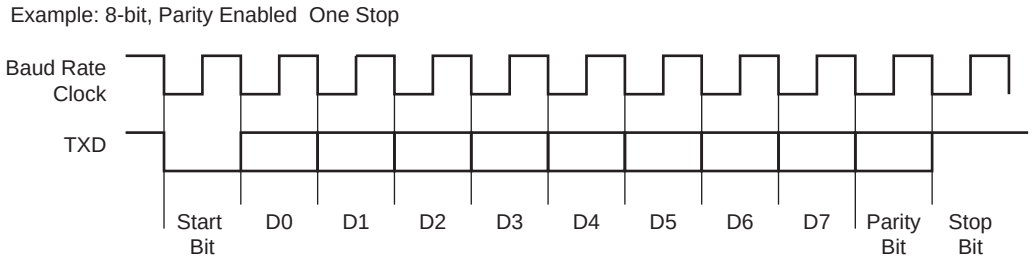
45.6.3.1 Transmitter Operations

The transmitter performs the same in both Synchronous and Asynchronous operating modes (SYNC = 0 or SYNC = 1). One start bit, up to 9 data bits, one optional parity bit and up to two stop bits are successively shifted out on the TXD pin at each falling edge of the programmed serial clock.

The number of data bits is selected by the CHRL field and the MODE 9 bit in US_MR. Nine bits are selected by setting the MODE 9 bit regardless of the CHRL field. The parity bit is set according to the PAR field in US_MR. The even, odd, space, marked or none parity bit can be configured. The MSBF field in the US_MR configures which data bit is sent first. If written to 1, the most significant bit is sent first. If written to 0, the less significant bit is sent first. The number of stop bits is selected by the NBSTOP field in the US_MR. The 1.5 stop bit is supported in Asynchronous mode only.

Downloaded from Arrow.com.

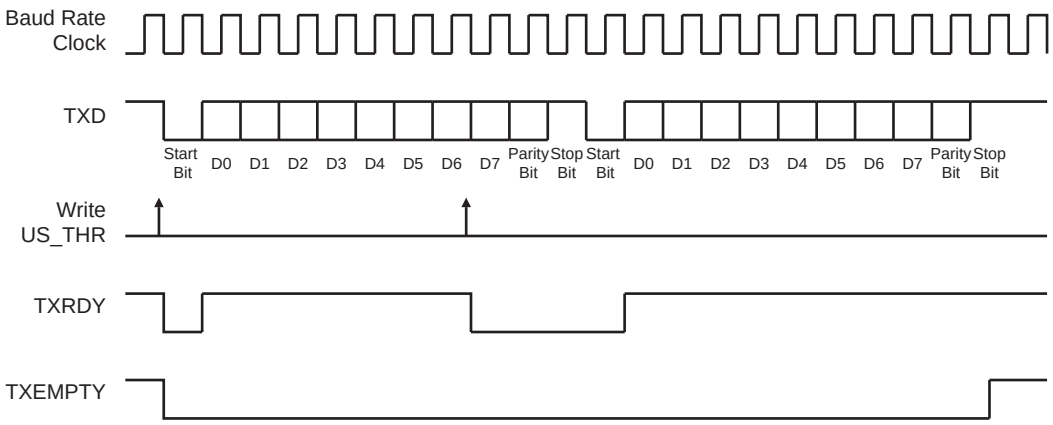
Figure 45-5. Character Transmit



The characters are sent by writing in the Transmit Holding register (US_THR). The transmitter reports two status bits in the Channel Status register (US_CSR): TXRDY (Transmitter Ready), which indicates that US_THR is empty and TXEMPTY, which indicates that all the characters written in US_THR have been processed. When the current character processing is completed, the last character written in US_THR is transferred into the Shift register of the transmitter and US_THR becomes empty, thus TXRDY rises.

Both TXRDY and TXEMPTY bits are low when the transmitter is disabled. Writing a character in US_THR while TXRDY is low has no effect and the written character is lost.

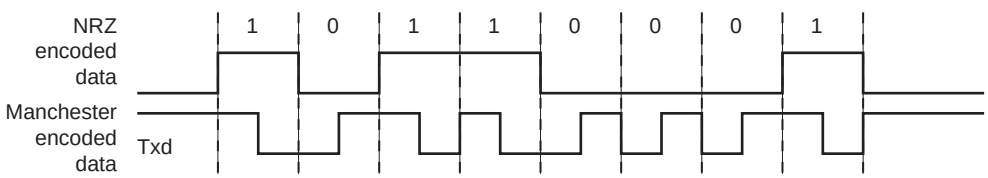
Figure 45-6. Transmitter Status



45.6.3.2 Manchester Encoder

When the Manchester encoder is in use, characters transmitted through the USART are encoded based on biphase Manchester II format. To enable this mode, set the MAN bit in the US_MR to 1. Depending on polarity configuration, a logic level (zero or one), is transmitted as a coded signal one-to-zero or zero-to-one. Thus, a transition always occurs at the midpoint of each bit time. It consumes more bandwidth than the original NRZ signal (2x) but the receiver has more error control since the expected input must show a change at the center of a bit cell. An example of Manchester encoded sequence is: the byte 0xB1 or 10110001 encodes to 10 01 10 10 01 01 01 10, assuming the default polarity of the encoder. Figure 45-7 illustrates this coding scheme.

Figure 45-7. NRZ to Manchester Encoding

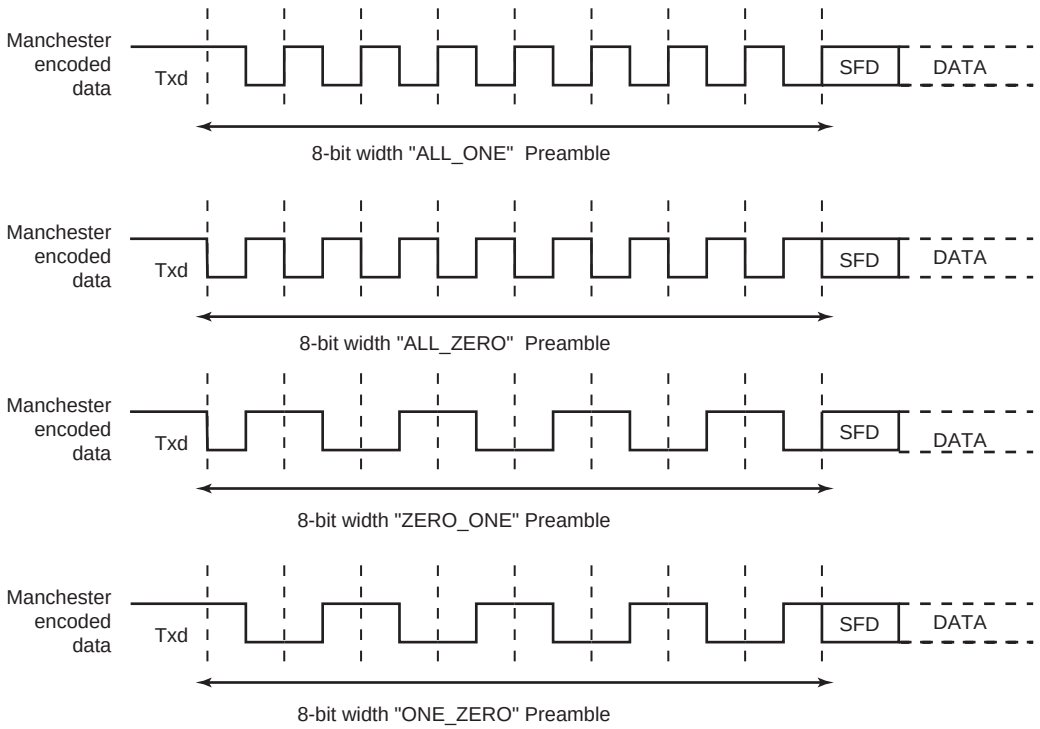


The Manchester encoded character can also be encapsulated by adding both a configurable preamble and a start frame delimiter pattern. Depending on the configuration, the preamble is a training sequence, composed of a

Downloaded from Arrow.com.

predefined pattern with a programmable length from 1 to 15 bit times. If the preamble length is set to 0, the preamble waveform is not generated prior to any character. The preamble pattern is chosen among the following sequences: ALL_ONE, ALL_ZERO, ONE_ZERO or ZERO_ONE, writing the field TX_PP in the US_MAN register, the field TX_PL is used to configure the preamble length. Figure 45-8 illustrates and defines the valid patterns. To improve flexibility, the encoding scheme can be configured using the TX_MPOL field in the US_MAN register. If the TX_MPOL field is set to zero (default), a logic zero is encoded with a zero-to-one transition and a logic one is encoded with a one-to-zero transition. If the TX_MPOL field is set to 1, a logic one is encoded with a one-to-zero transition and a logic zero is encoded with a zero-to-one transition.

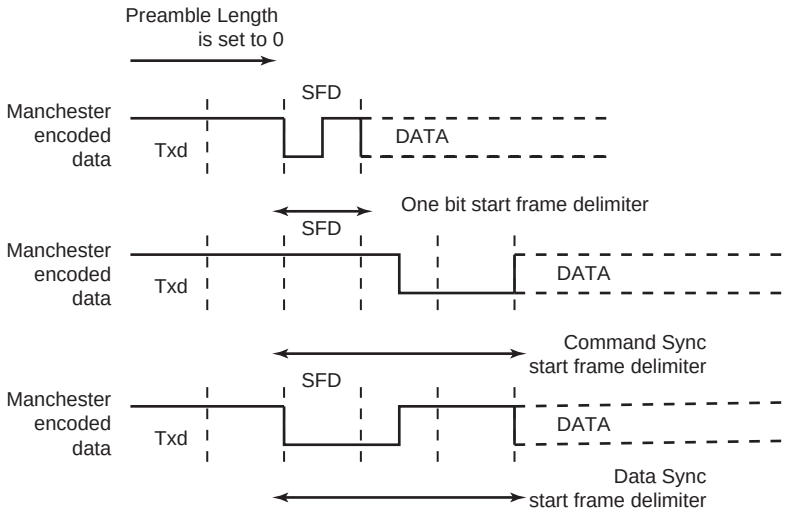
Figure 45-8. Preamble Patterns, Default Polarity Assumed



A start frame delimiter is to be configured using the ONEBIT bit in the US_MR. It consists of a user-defined pattern that indicates the beginning of a valid data. Figure 45-9 illustrates these patterns. If the start frame delimiter, also known as the start bit, is one bit, (ONEBIT = 1), a logic zero is Manchester encoded and indicates that a new character is being sent serially on the line. If the start frame delimiter is a synchronization pattern also referred to as sync (ONEBIT to 0), a sequence of three bit times is sent serially on the line to indicate the start of a new character. The sync waveform is in itself an invalid Manchester waveform as the transition occurs at the middle of the second bit time. Two distinct sync patterns are used: the command sync and the data sync. The command sync has a logic one level for one and a half bit times, then a transition to logic zero for the second one and a half bit times. If the MODSYNC bit in the US_MR is set to 1, the next character is a command. If it is set to 0, the next character is a data. When direct memory access is used, the MODSYNC field can be immediately updated with a modified character located in memory. To enable this mode, VAR_SYNC bit in US_MR must be set to 1. In this case, the MODSYNC bit in the US_MR is bypassed and the sync configuration is held in the TXSYNH in the US_THR. The USART character format is modified and includes sync information.

Downloaded from Arrow.com.

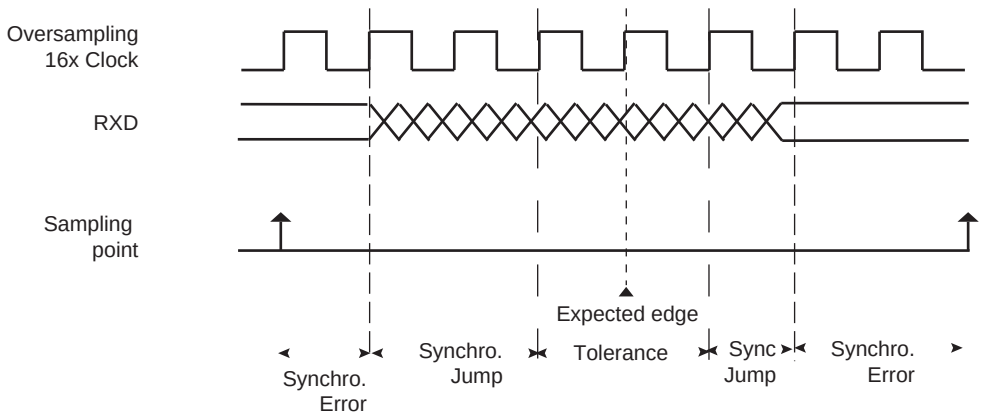
Figure 45-9. Start Frame Delimiter



Drift Compensation

Drift compensation is available only in 16X Oversampling mode. An hardware recovery system allows a larger clock drift. To enable the hardware system, the bit in the USART_MAN register must be set. If the RXD edge is one 16X clock cycle from the expected edge, this is considered as normal jitter and no corrective actions is taken. If the RXD event is between 4 and 2 clock cycles before the expected edge, then the current period is shortened by one clock cycle. If the RXD event is between 2 and 3 clock cycles after the expected edge, then the current period is lengthened by one clock cycle. These intervals are considered to be drift and so corrective actions are automatically taken.

Figure 45-10. Bit Resynchronization



45.6.3.3 Asynchronous Receiver

If the USART is programmed in Asynchronous operating mode (SYNC = 0), the receiver oversamples the RXD input line. The oversampling is either 16 or 8 times the baud rate clock, depending on the OVER bit in the US_MR.

The receiver samples the RXD line. If the line is sampled during one half of a bit time to 0, a start bit is detected and data, parity and stop bits are successively sampled on the bit rate clock.

If the oversampling is 16 (OVER = 0), a start is detected at the eighth sample to 0. Data bits, parity bit and stop bit are assumed to have a duration corresponding to 16 oversampling clock cycles. If the oversampling is 8 (OVER = 1), a start bit is detected at the fourth sample to 0. Data bits, parity bit and stop bit are assumed to have a duration corresponding to 8 oversampling clock cycles.

Downloaded from Arrow.com.

The number of data bits, first bit sent and Parity mode are selected by the same fields and bits as the transmitter, i.e., respectively CHRL, MODE9, MSBF and PAR. For the synchronization mechanism **only**, the number of stop bits has no effect on the receiver as it considers only one stop bit, regardless of the field NBSTOP, so that resynchronization between the receiver and the transmitter can occur. Moreover, as soon as the stop bit is sampled, the receiver starts looking for a new start bit so that resynchronization can also be accomplished when the transmitter is operating with one stop bit.

Figure 45-11 and Figure 45-12 illustrate start detection and character reception when USART operates in Asynchronous mode.

Figure 45-11. Asynchronous Start Detection

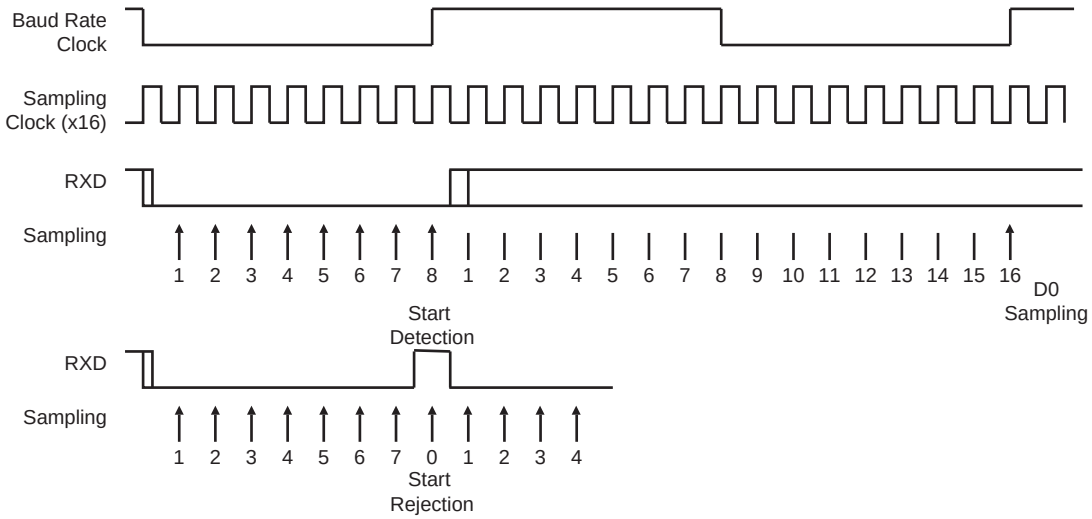
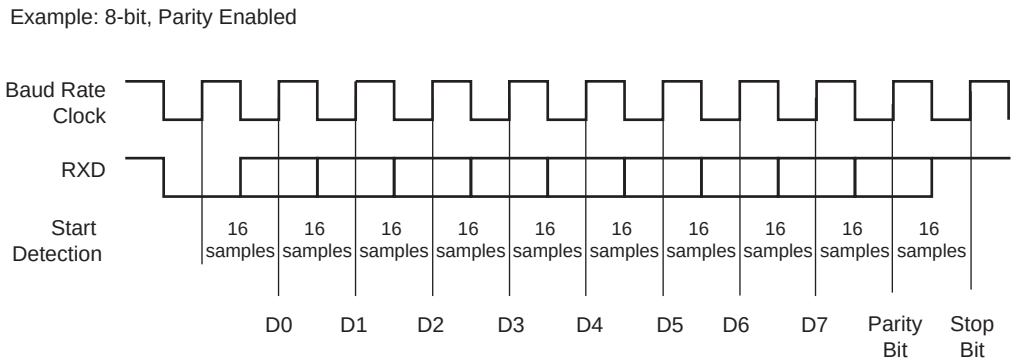


Figure 45-12. Asynchronous Character Reception



45.6.3.4 Manchester Decoder

When the MAN bit in the US_MR is set to 1, the Manchester decoder is enabled. The decoder performs both preamble and start frame delimiter detection. One input line is dedicated to Manchester encoded input data.

An optional preamble sequence can be defined, its length is user-defined and totally independent of the emitter side. Use RX_PL in US_MAN register to configure the length of the preamble sequence. If the length is set to 0, no preamble is detected and the function is disabled. In addition, the polarity of the input stream is programmable with RX_MPOL bit in US_MAN register. Depending on the desired application the preamble pattern matching is to be defined via the RX_PP field in US_MAN. See Figure 45-8 for available preamble patterns.

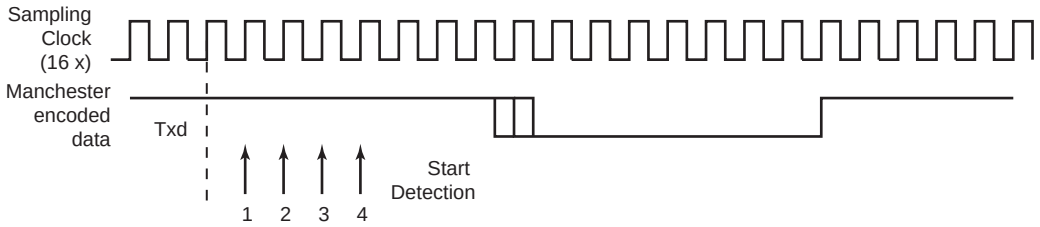
Unlike preamble, the start frame delimiter is shared between Manchester Encoder and Decoder. So, if ONEBIT field is set to 1, only a zero encoded Manchester can be detected as a valid start frame delimiter. If ONEBIT is set

Downloaded from Arrow.com.

to 0, only a sync pattern is detected as a valid start frame delimiter. Decoder operates by detecting transition on incoming stream. If RXD is sampled during one quarter of a bit time to zero, a start bit is detected. See Figure 45-13. The sample pulse rejection mechanism applies.

The RXIDLEV bit in the US_MAN informs the USART of the receiver line idle state value (receiver line inactive). The user must define RXIDLEV to ensure reliable synchronization. By default, RXIDLEV is set to 1 (receiver line is at level 1 when there is no activity).

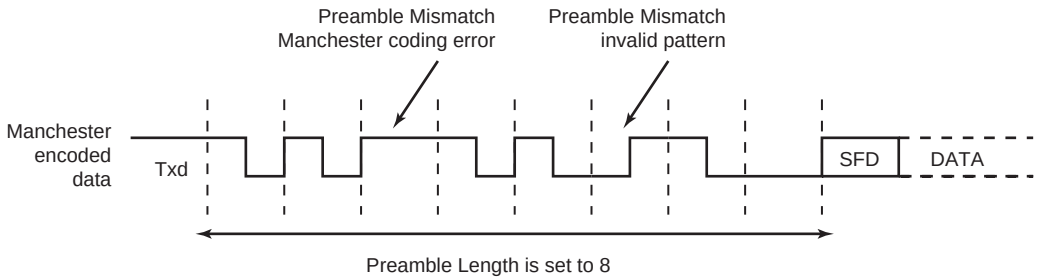
Figure 45-13. Asynchronous Start Bit Detection



The receiver is activated and starts preamble and frame delimiter detection, sampling the data at one quarter and then three quarters. If a valid preamble pattern or start frame delimiter is detected, the receiver continues decoding with the same synchronization. If the stream does not match a valid pattern or a valid start frame delimiter, the receiver resynchronizes on the next valid edge.The minimum time threshold to estimate the bit value is three quarters of a bit time.

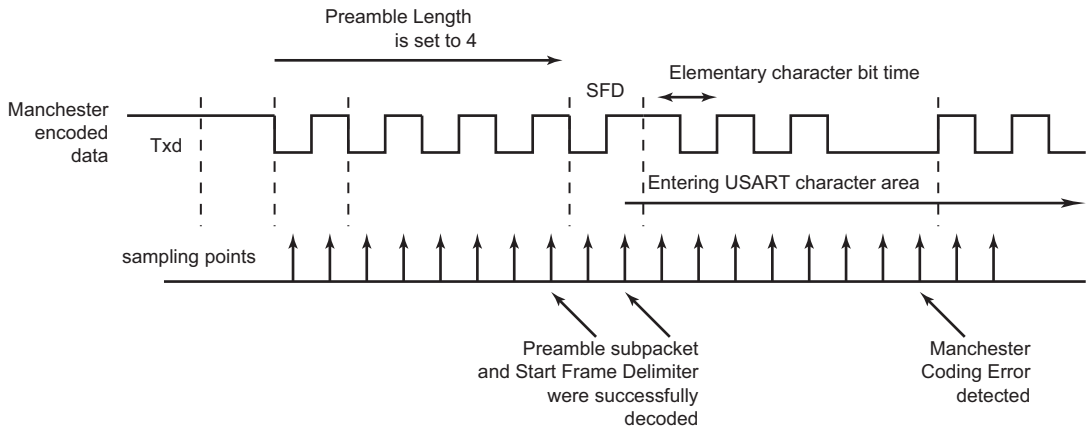
If a valid preamble (if used) followed with a valid start frame delimiter is detected, the incoming stream is decoded into NRZ data and passed to USART for processing. Figure 45-14 illustrates Manchester pattern mismatch. When incoming data stream is passed to the USART, the receiver is also able to detect Manchester code violation. A code violation is a lack of transition in the middle of a bit cell. In this case, the MANERR flag in the US_CSR is raised. It is cleared by writing a 1 to the RSTSTA in the US_CR. See Figure 45-15 for an example of Manchester error detection during data phase.

Figure 45-14. Preamble Pattern Mismatch



Downloaded from Arrow.com.

Figure 45-15. Manchester Error Flag



When the start frame delimiter is a sync pattern (ONEBIT field to 0), both command and data delimiter are supported. If a valid sync is detected, the received character is written as RXCHR field in the US_RHR and the RXSYNH is updated. RXCHR is set to 1 when the received character is a command, and it is set to 0 if the received character is a data. This mechanism alleviates and simplifies the direct memory access as the character contains its own sync field in the same register.

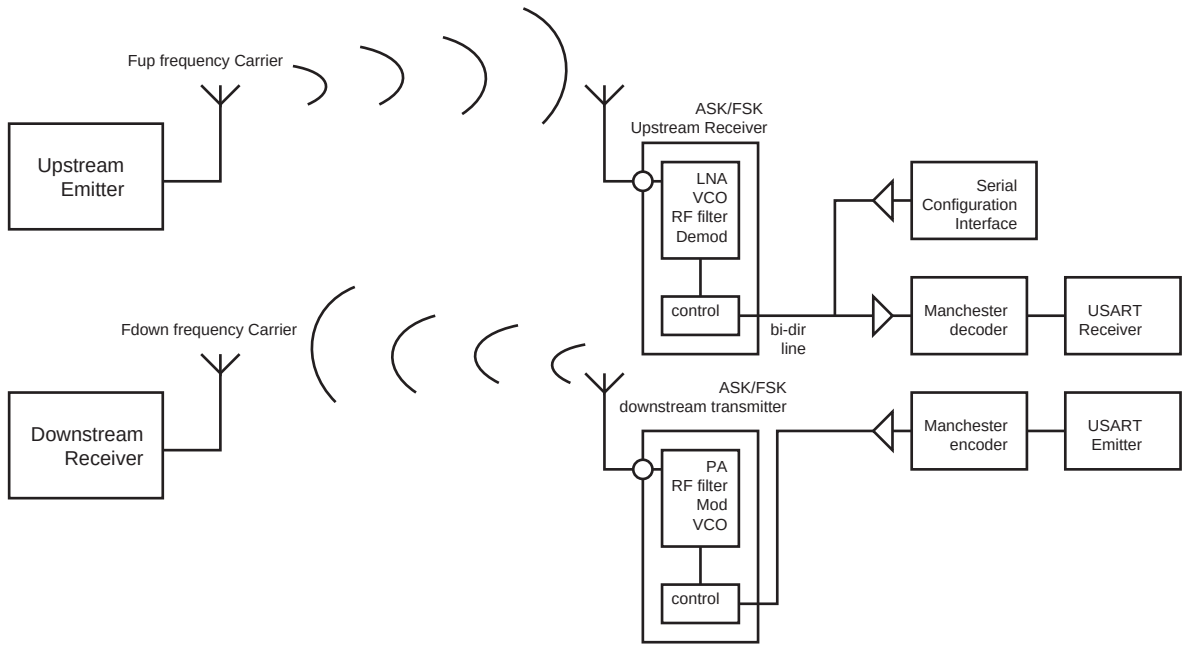
As the decoder is setup to be used in Unipolar mode, the first bit of the frame has to be a zero-to-one transition.

45.6.3.5 Radio Interface: Manchester Encoded USART Application

This section describes low data rate RF transmission systems and their integration with a Manchester encoded USART. These systems are based on transmitter and receiver ICs that support ASK and FSK modulation schemes.

The goal is to perform full duplex radio transmission of characters using two different frequency carriers. See the configuration in Figure 45-16.

Figure 45-16. Manchester Encoded Characters RF Transmission



Downloaded from Arrow.com.

The USART peripheral is configured as a Manchester encoder/decoder. Looking at the downstream communication channel, Manchester encoded characters are serially sent to the RF emitter. This may also include a user defined preamble and a start frame delimiter. Mostly, preamble is used in the RF receiver to distinguish between a valid data from a transmitter and signals due to noise. The Manchester stream is then modulated. See Figure 45-17 for an example of ASK modulation scheme. When a logic one is sent to the ASK modulator, the power amplifier, referred to as PA, is enabled and transmits an RF signal at downstream frequency. When a logic zero is transmitted, the RF signal is turned off. If the FSK modulator is activated, two different frequencies are used to transmit data. When a logic 1 is sent, the modulator outputs an RF signal at frequency F0 and switches to F1 if the data sent is a 0. See Figure 45-18.

From the receiver side, another carrier frequency is used. The RF receiver performs a bit check operation examining demodulated data stream. If a valid pattern is detected, the receiver switches to Receiving mode. The demodulated stream is sent to the Manchester decoder. Because of bit checking inside RF IC, the data transferred to the microcontroller is reduced by a user-defined number of bits. The Manchester preamble length is to be defined in accordance with the RF IC configuration.

Figure 45-17. ASK Modulator Output

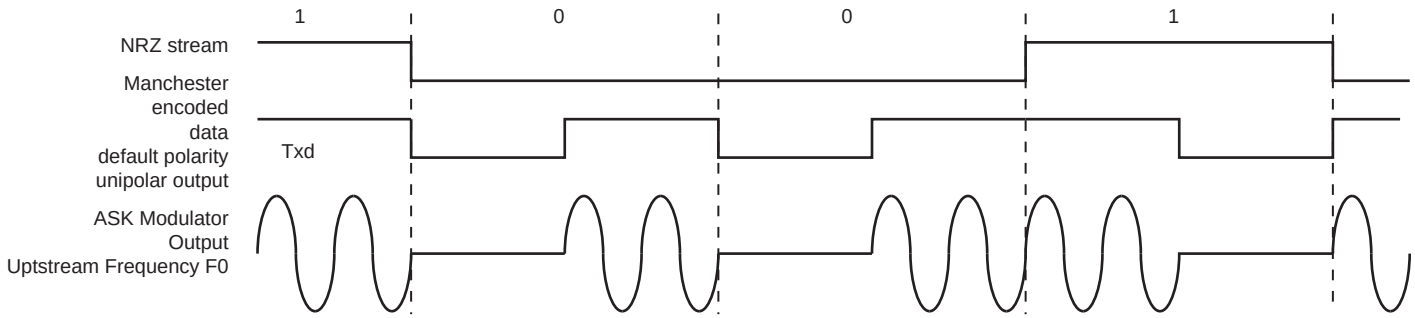
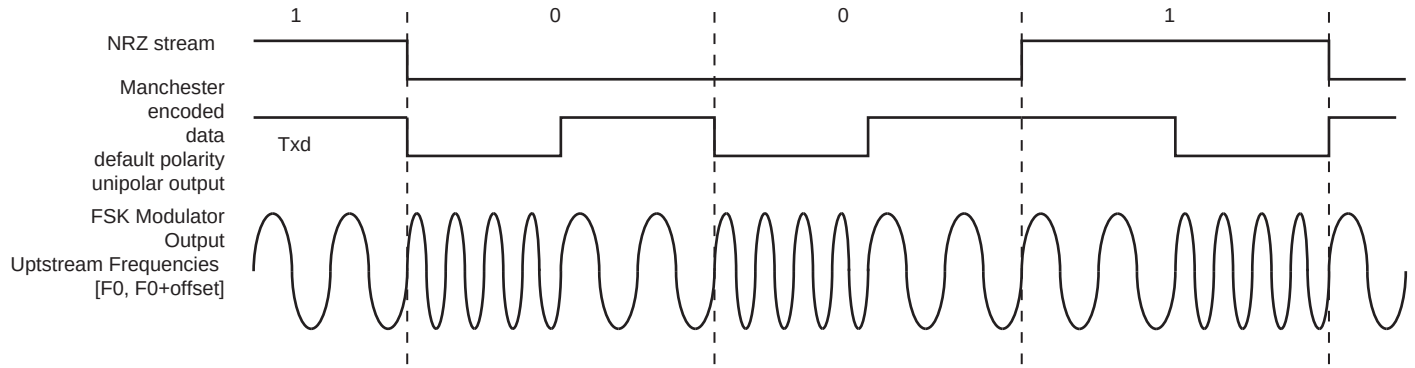


Figure 45-18. FSK Modulator Output



45.6.3.6 Synchronous Receiver

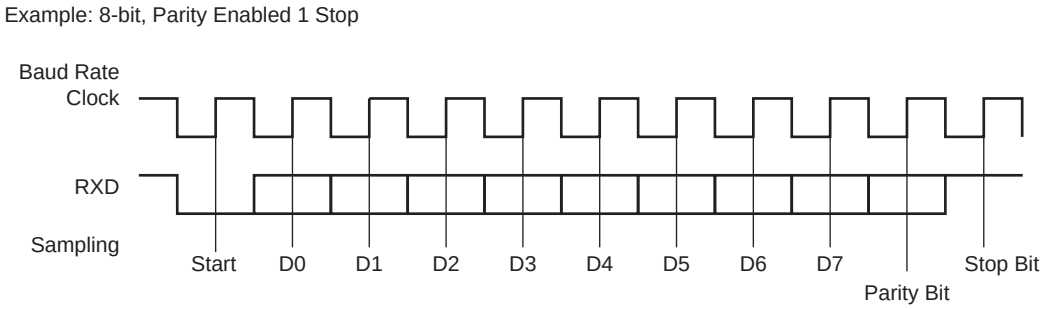
In Synchronous mode (SYNC = 1), the receiver samples the RXD signal on each rising edge of the baud rate clock. If a low level is detected, it is considered as a start. All data bits, the parity bit and the stop bits are sampled and the receiver waits for the next start bit. Synchronous mode operations provide a high-speed transfer capability.

Configuration fields and bits are the same as in Asynchronous mode.

Figure 45-19 illustrates a character reception in Synchronous mode.

Downloaded from Arrow.com.

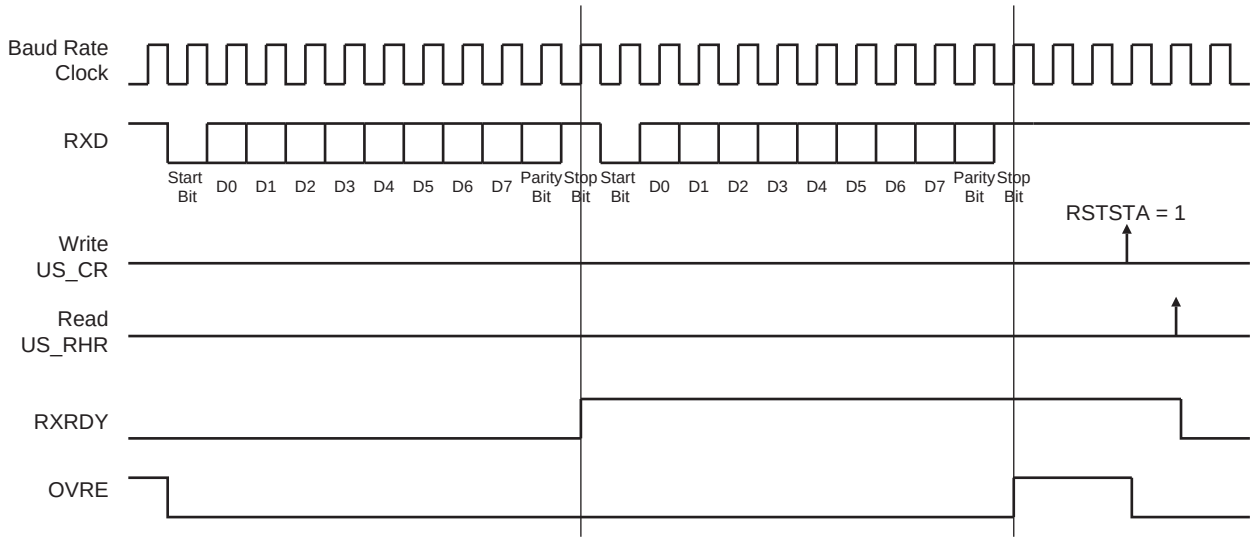
Figure 45-19. Synchronous Mode Character Reception



45.6.3.7 Receiver Operations

When a character reception is completed, it is transferred to the Receive Holding register (US_RHR) and the RXRDY bit in US_CSR rises. If a character is completed while the RXRDY is set, the OVRE (Overrun Error) bit is set. The last character is transferred into US_RHR and overwrites the previous one. The OVRE bit is cleared by writing a 1 to the RSTSTA (Reset Status) bit in the US_CR.

Figure 45-20. Receiver Status



45.6.3.8 Parity

The USART supports five Parity modes that are selected by writing to the PAR field in the US_MR. The PAR field also enables the Multidrop mode, see Section 45.6.3.9 ”Multidrop Mode”. Even and odd parity bit generation and error detection are supported.

If even parity is selected, the parity generator of the transmitter drives the parity bit to 0 if a number of 1s in the character data bit is even, and to 1 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If odd parity is selected, the parity generator of the transmitter drives the parity bit to 1 if a number of 1s in the character data bit is even, and to 0 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If the mark parity is used, the parity generator of the transmitter drives the parity bit to 1 for all characters. The receiver parity checker reports an error if the parity bit is sampled to 0. If the space parity is used, the parity generator of the transmitter drives the parity bit to 0 for all characters. The receiver parity checker reports an error if the parity bit is sampled to 1. If parity is disabled, the transmitter does not generate any parity bit and the receiver does not report any parity error.

Downloaded from Arrow.com.

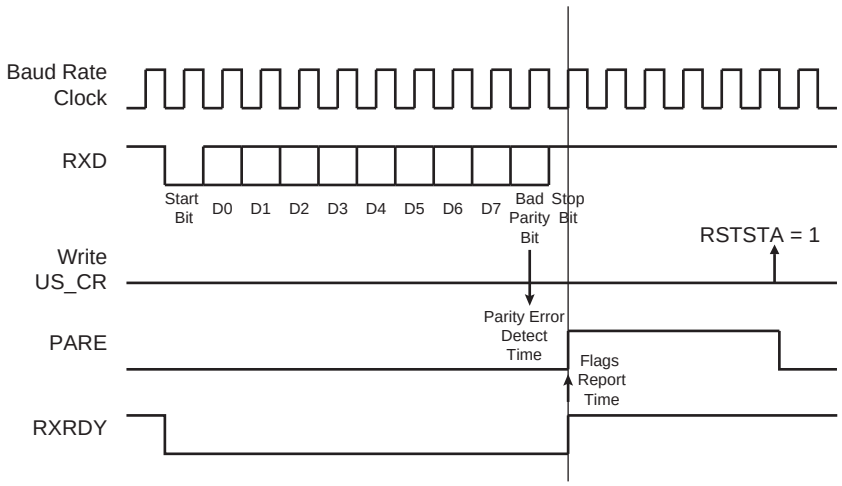
Table 45-8 shows an example of the parity bit for the character 0x41 (character ASCII “A”) depending on the configuration of the USART. Because there are two bits set to 1 in the character value, the parity bit is set to 1 when the parity is odd, or configured to 0 when the parity is even.

Table 45-8. Parity Bit Examples

Character	Hexadecimal	Binary	Parity Bit	Parity Mode
A	0x41	0100 0001	1	Odd
A	0x41	0100 0001	0	Even
A	0x41	0100 0001	1	Mark
A	0x41	0100 0001	0	Space
A	0x41	0100 0001	None	None

When the receiver detects a parity error, it sets the PARE (Parity Error) bit in the US_CSR. The PARE bit can be cleared by writing a 1 to the RSTSTA bit the US_CR. Figure 45-21 illustrates the parity bit status setting and clearing.

Figure 45-21. Parity Error



45.6.3.9 Multidrop Mode

If the value 0x6 or 0x07 is written to the PAR field in the US_MR, the USART runs in Multidrop mode. This mode differentiates the data characters and the address characters. Data is transmitted with the parity bit at 0 and addresses are transmitted with the parity bit at 1.

If the USART is configured in Multidrop mode, the receiver sets the PARE parity error bit when the parity bit is high and the transmitter is able to send a character with the parity bit high when a 1 is written to the SENTA bit in the US_CR.

To handle parity error, the PARE bit is cleared when a 1 is written to the RSTSTA bit in the US_CR.

The transmitter sends an address byte (parity bit set) when SENDA is written to in the US_CR. In this case, the next byte written to the US_THR is transmitted as an address. Any character written in the US_THR without having written the command SENDA is transmitted normally with the parity at 0.

45.6.3.10 Transmitter Timeguard

The timeguard feature enables the USART interface with slow remote devices.

The timeguard function enables the transmitter to insert an idle state on the TXD line between two characters. This idle state actually acts as a long stop bit.

Downloaded from Arrow.com.

The duration of the idle state is programmed in the TG field of the Transmitter Timeguard register (US_TTGR). When this field is written to zero no timeguard is generated. Otherwise, the transmitter holds a high level on TXD after each transmitted byte during the number of bit periods programmed in TG in addition to the number of stop bits.

As illustrated in Figure 45-22, the behavior of TXRDY and TXEMPTY status bits is modified by the programming of a timeguard. TXRDY rises only when the start bit of the next character is sent, and thus remains to 0 during the timeguard transmission if a character has been written in US_THR. TXEMPTY remains low until the timeguard transmission is completed as the timeguard is part of the current character being transmitted.

Figure 45-22. Timeguard Operations

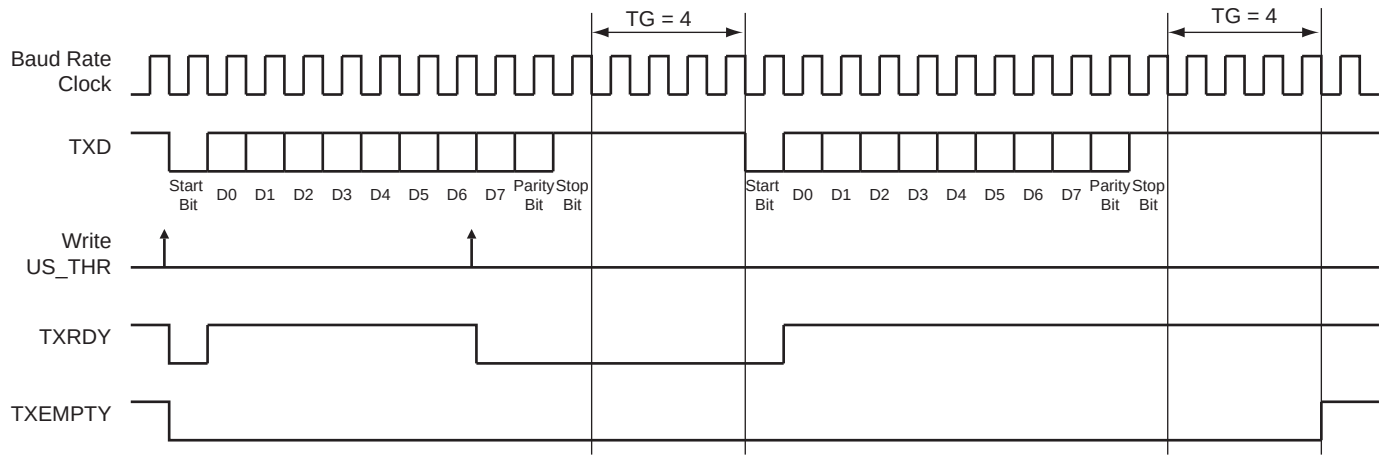


Table 45-9 indicates the maximum length of a timeguard period that the transmitter can handle in relation to the function of the baud rate.

Table 45-9. Maximum Timeguard Length Depending on Baud Rate

Baud Rate (bit/s)	Bit Time (µs)	Timeguard (ms)
1,200	833	212.50
9,600	104	26.56
14,400	69.4	17.71
19,200	52.1	13.28
28,800	34.7	8.85
38,400	26	6.63
56,000	17.9	4.55
57,600	17.4	4.43
115,200	8.7	2.21

45.6.3.11 Receiver Time-out

The Receiver Time-out provides support in handling variable-length frames. This feature detects an idle condition on the RXD line. When a time-out is detected, the bit TIMEOUT in the US_CSR rises and can generate an interrupt, thus indicating to the driver an end of frame.

The time-out delay period (during which the receiver waits for a new character) is programmed in the TO field of the Receiver Time-out register (US_RTOR). If the TO field is written to 0, the Receiver Time-out is disabled and no time-out is detected. The TIMEOUT bit in the US_CSR remains at 0. Otherwise, the receiver loads a 16-bit counter

Downloaded from Arrow.com.

with the value programmed in TO. This counter is decremented at each bit period and reloaded each time a new character is received. If the counter reaches 0, the TIMEOUT bit in US_CSR rises. Then, the user can either:

- Stop the counter clock until a new character is received. This is performed by writing a 1 to the STTTO (Start Time-out) bit in the US_CR. In this case, the idle state on RXD before a new character is received will not provide a time-out. This prevents having to handle an interrupt before a character is received and allows waiting for the next idle state on RXD after a frame is received.
- Obtain an interrupt while no character is received. This is performed by writing a 1 to the RETTO (Reload and Start Time-out) bit in the US_CR. If RETTO is performed, the counter starts counting down immediately from the value TO. This enables generation of a periodic interrupt so that a user time-out can be handled, for example when no key is pressed on a keyboard.

If STTTO is performed, the counter clock is stopped until a first character is received. The idle state on RXD before the start of the frame does not provide a time-out. This prevents having to obtain a periodic interrupt and enables a wait of the end of frame when the idle state on RXD is detected.

If RETTO is performed, the counter starts counting down immediately from the value TO. This enables generation of a periodic interrupt so that a user time-out can be handled, for example when no key is pressed on a keyboard.

Figure 45-23 shows the block diagram of the Receiver Time-out feature.

Figure 45-23. Receiver Time-out Block Diagram

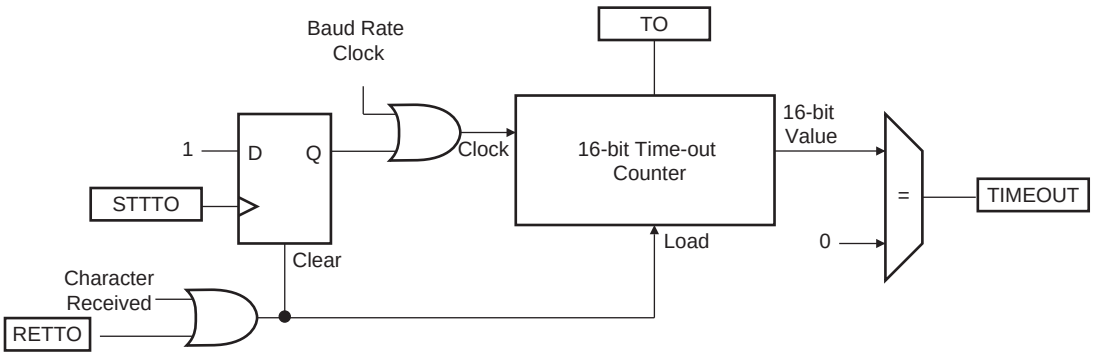


Table 45-10 gives the maximum time-out period for some standard baud rates.

Table 45-10. Maximum Time-out Period

Baud Rate (bit/s)	Bit Time (µs)	Time-out (ms)
600	1,667	109,225
1,200	833	54,613
2,400	417	27,306
4,800	208	13,653
9,600	104	6,827
14,400	69	4,551
19,200	52	3,413
28,800	35	2,276
38,400	26	1,704
56,000	18	1,170
57,600	17	1,138
200,000	5	328

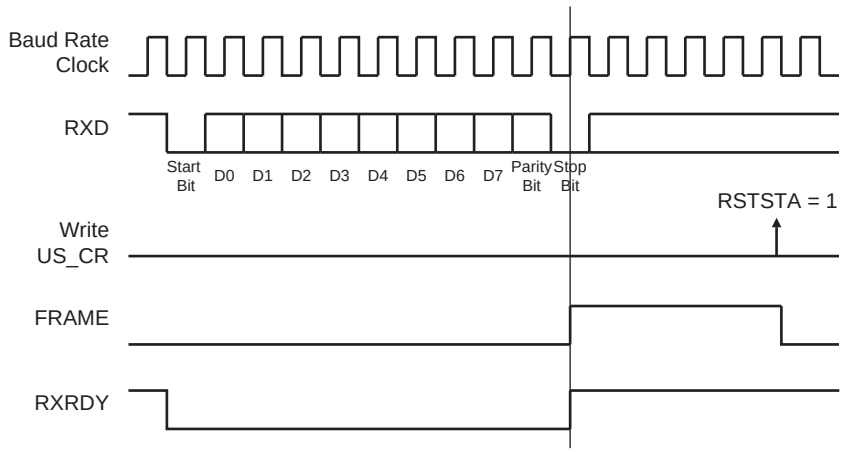
Downloaded from Arrow.com.

45.6.3.12 Framing Error

The receiver is capable of detecting framing errors. A framing error happens when the stop bit of a received character is detected at level 0. This can occur if the receiver and the transmitter are fully desynchronized.

A framing error is reported on the FRAME bit of US_CSR. The FRAME bit is asserted in the middle of the stop bit as soon as the framing error is detected. It is cleared by writing a 1 to the RSTSTA bit in the US_CR.

Figure 45-24. Framing Error Status



45.6.3.13 Transmit Break

The user can request the transmitter to generate a break condition on the TXD line. A break condition drives the TXD line low during at least one complete character. It appears the same as a 0x00 character sent with the parity and the stop bits at 0. However, the transmitter holds the TXD line at least during one character until the user requests the break condition to be removed.

A break is transmitted by writing a 1 to the STTBRK bit in the US_CR. This can be performed at any time, either while the transmitter is empty (no character in either the Shift register or in US_THR) or when a character is being transmitted. If a break is requested while a character is being shifted out, the character is first completed before the TXD line is held low.

Once STTBRK command is requested further STTBRK commands are ignored until the end of the break is completed.

The break condition is removed by writing a 1 to the STPBRK bit in the US_CR. If the STPBRK is requested before the end of the minimum break duration (one character, including start, data, parity and stop bits), the transmitter ensures that the break condition completes.

The transmitter considers the break as though it is a character, i.e., the STTBRK and STPBRK commands are processed only if the TXRDY bit in US_CSR is to 1 and the start of the break condition clears the TXRDY and TXEMPTY bits as if a character is processed.

Writing US_CR with both STTBRK and STPBRK bits to 1 can lead to an unpredictable result. All STPBRK commands requested without a previous STTBRK command are ignored. A byte written into the Transmit Holding register while a break is pending, but not started, is ignored.

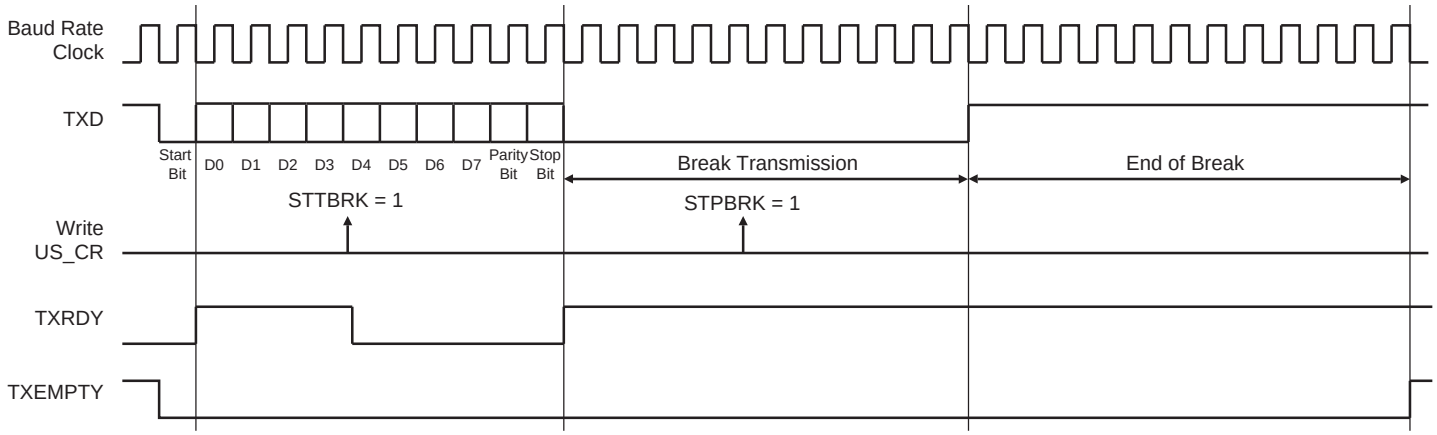
After the break condition, the transmitter returns the TXD line to 1 for a minimum of 12 bit times. Thus, the transmitter ensures that the remote receiver detects correctly the end of break and the start of the next character. If the timeguard is programmed with a value higher than 12, the TXD line is held high for the timeguard period.

After holding the TXD line for this period, the transmitter resumes normal operations.

Figure 45-25 illustrates the effect of both the Start Break (STTBRK) and Stop Break (STPBRK) commands on the TXD line.

Downloaded from Arrow.com.

Figure 45-25. Break Transmission



45.6.3.14 Receive Break

The receiver detects a break condition when all data, parity and stop bits are low. This corresponds to detecting a framing error with data to 0x00, but FRAME remains low.

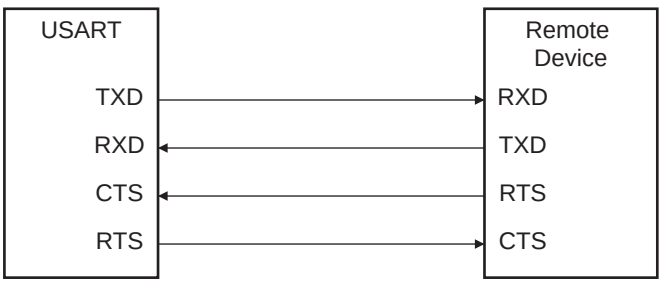
When the low stop bit is detected, the receiver asserts the RXBRK bit in US_CSR. This bit may be cleared by writing a 1 to the RSTSTA bit in the US_CR.

An end of receive break is detected by a high level for at least 2/16 of a bit period in Asynchronous operating mode or one sample at high level in Synchronous operating mode. The end of break detection also asserts the RXBRK bit.

45.6.3.15 Hardware Handshaking

The USART features a hardware handshaking out-of-band flow control. The RTS and CTS pins are used to connect with the remote device, as shown in Figure 45-26.

Figure 45-26. Connection with a Remote Device for Hardware Handshaking



Setting the USART to operate with hardware handshaking is performed by writing the USART_MODE field in US_MR to the value 0x2.

The USART behavior when hardware handshaking is enabled is the same as the behavior in standard Synchronous or Asynchronous mode, except that the receiver drives the RTS pin as described below and the level on the CTS pin modifies the behavior of the transmitter as described below. The transmitter can handle hardware handshaking in any case.

Downloaded from Arrow.com.

Figure 45-27. RTS Line Software Control when USART_MR.USART_MODE = 2

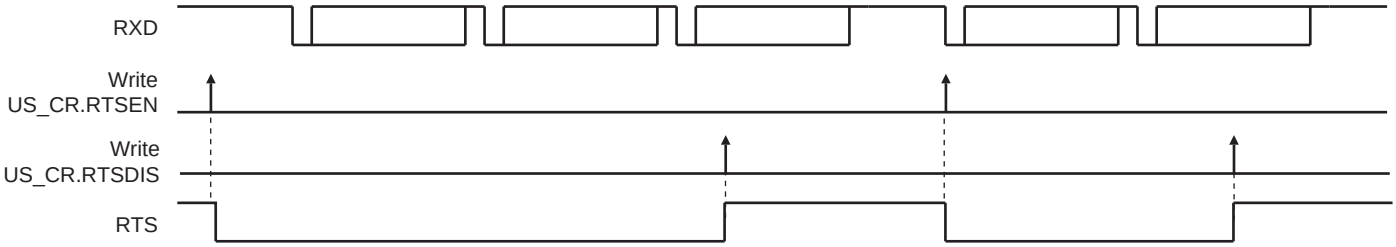
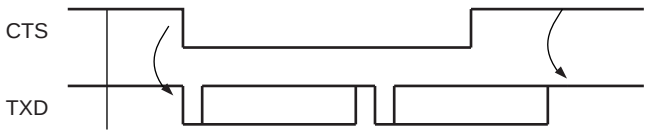


Figure 45-28 shows how the transmitter operates if hardware handshaking is enabled. The CTS pin disables the transmitter. If a character is being processed, the transmitter is disabled only after the completion of the current character and transmission of the next character happens as soon as the pin CTS falls.

Figure 45-28. Transmitter Behavior when Operating with Hardware Handshaking



45.6.4 ISO7816 Mode

The USART features an ISO7816-compatible operating mode. This mode permits interfacing with smart cards and Security Access Modules (SAM) communicating through an ISO7816 link. Both T = 0 and T = 1 protocols defined by the ISO7816 specification are supported.

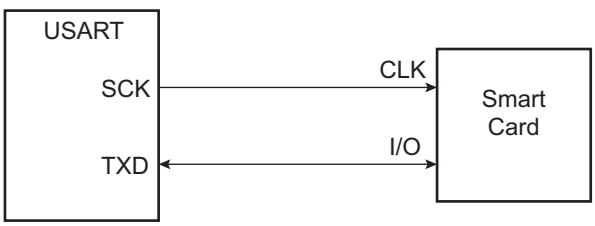
Setting the USART in ISO7816 mode is performed by writing the USART_MODE field in US_MR to the value 0x4 for protocol T = 0 and to the value 0x5 for protocol T = 1.

45.6.4.1 ISO7816 Mode Overview

The ISO7816 is a half duplex communication on only one bidirectional line. The baud rate is determined by a division of the clock provided to the remote device (see Section 45-2 ”Baud Rate Generator”).

The USART connects to a smart card as shown in Figure 45-29. The TXD line becomes bidirectional and the baud rate generator feeds the ISO7816 clock on the SCK pin. As the TXD pin becomes bidirectional, its output remains driven by the output of the transmitter but only when the transmitter is active while its input is directed to the input of the receiver. The USART is considered as the master of the communication as it generates the clock.

Figure 45-29. Connection of a Smart Card to the USART



When operating in ISO7816, either in T = 0 or T = 1 modes, the character format is fixed. The configuration is 8 data bits, even parity and 1 or 2 stop bits, regardless of the values programmed in the CHRL, MODE9, PAR and CHMODE fields. MSBF can be used to transmit LSB or MSB first. Parity Bit (PAR) can be used to transmit in Normal or Inverse mode. Refer to Section 45.7.3 ”USART Mode Register” and “PAR: Parity Type” .

The USART cannot operate concurrently in both Receiver and Transmitter modes as the communication is unidirectional at a time. It has to be configured according to the required mode by enabling or disabling either the

Downloaded from Arrow.com.

receiver or the transmitter as desired. Enabling both the receiver and the transmitter at the same time in ISO7816 mode may lead to unpredictable results.

The ISO7816 specification defines an inverse transmission format. Data bits of the character must be transmitted on the I/O line at their negative value.

45.6.4.2 Protocol T = 0

In T = 0 protocol, a character is made up of one start bit, eight data bits, one parity bit and one guard time, which lasts two bit times. The transmitter shifts out the bits and does not drive the I/O line during the guard time.

If no parity error is detected, the I/O line remains at 1 during the guard time and the transmitter can continue with the transmission of the next character, as shown in Figure 45-30.

If a parity error is detected by the receiver, it drives the I/O line to 0 during the guard time, as shown in Figure 45-31. This error bit is also named NACK, for Non Acknowledge. In this case, the character lasts 1 bit time more, as the guard time length is the same and is added to the error bit time which lasts 1 bit time.

When the USART is the receiver and it detects an error, it does not load the erroneous character in the Receive Holding register (US_RHR). It appropriately sets the PARE bit in the Status register (US_SR) so that the software can handle the error.

Figure 45-30. T = 0 Protocol without Parity Error

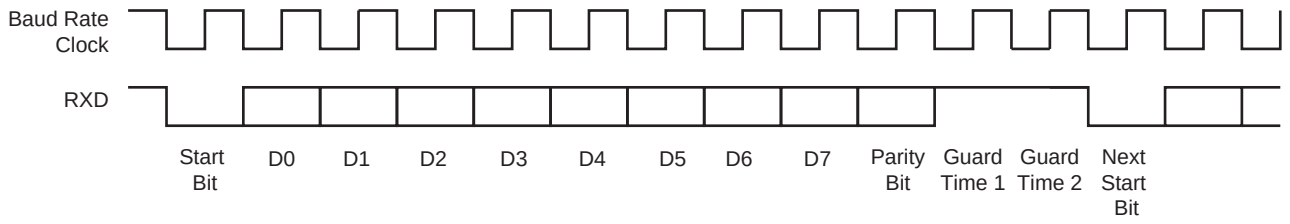
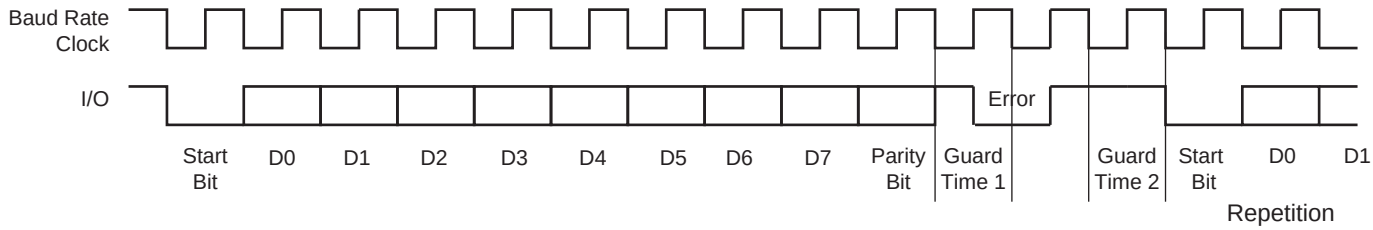


Figure 45-31. T = 0 Protocol with Parity Error



Receive Error Counter

The USART receiver also records the total number of errors. This can be read in the Number of Error (US_NER) register. The NB_ERRORS field can record up to 255 errors. Reading US_NER automatically clears the NB_ERRORS field.

Receive NACK Inhibit

The USART can also be configured to inhibit an error. This can be achieved by setting the INACK bit in US_MR. If INACK is to 1, no error signal is driven on the I/O line even if a parity bit is detected.

Moreover, if INACK is set, the erroneous received character is stored in the Receive Holding register, as if no error occurred and the RXRDY bit does rise.

Transmit Character Repetition

When the USART is transmitting a character and gets a NACK, it can automatically repeat the character before moving on to the next one. Repetition is enabled by writing the MAX_ITERATION field in the US_MR at a value higher than 0. Each character can be transmitted up to eight times; the first transmission plus seven repetitions.

Downloaded from Arrow.com.

If MAX_ITERATION does not equal zero, the USART repeats the character as many times as the value loaded in MAX_ITERATION.

When the USART repetition number reaches MAX_ITERATION and the last repeated character is not acknowledged, the ITER bit is set in US_CSR. If the repetition of the character is acknowledged by the receiver, the repetitions are stopped and the iteration counter is cleared.

The ITER bit in US_CSR can be cleared by writing a 1 to the RSTIT bit in the US_CR.

Disable Successive Receive NACK

The receiver can limit the number of successive NACKs sent back to the remote transmitter. This is programmed by setting the bit DSNACK in the US_MR. The maximum number of NACKs transmitted is programmed in the MAX_ITERATION field. As soon as MAX_ITERATION is reached, no error signal is driven on the I/O line and the ITER bit in the US_CSR is set.

45.6.4.3 Protocol T = 1

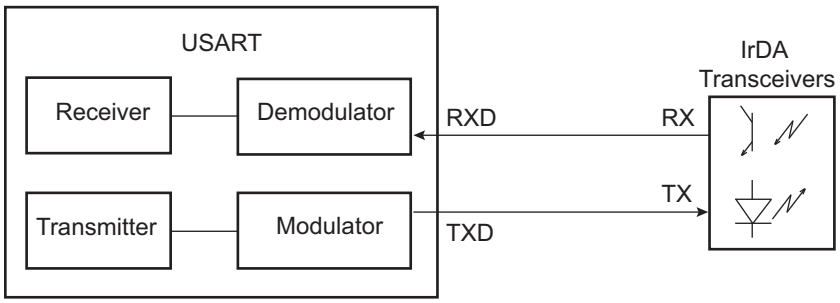
When operating in ISO7816 protocol T = 1, the transmission is similar to an asynchronous format with only one stop bit. The parity is generated when transmitting and checked when receiving. Parity error detection sets the PARE bit in the US_CSR.

45.6.5 IrDA Mode

The USART features an IrDA mode supplying half-duplex point-to-point wireless communication. It embeds the modulator and demodulator which allows a glueless connection to the infrared transceivers, as shown in Figure 45-32. The modulator and demodulator are compliant with the IrDA specification version 1.1 and support data transfer speeds ranging from 2.4 kbit/s to 115.2 kbit/s.

The IrDA mode is enabled by setting the USART_MODE field in US_MR to the value 0x8. The IrDA Filter register (US_IF) is used to configure the demodulator filter. The USART transmitter and receiver operate in a normal Asynchronous mode and all parameters are accessible. Note that the modulator and the demodulator are activated.

Figure 45-32. Connection to IrDA Transceivers



The receiver and the transmitter must be enabled or disabled depending on the direction of the transmission to be managed.

To receive IrDA signals, the following needs to be done:

- Disable TX and Enable RX
- Configure the TXD pin as PIO and set it as an output to 0 (to avoid LED emission). Disable the internal pull-up (better for power consumption).
- Receive data

Downloaded from Arrow.com.

45.6.5.1 IrDA Modulation

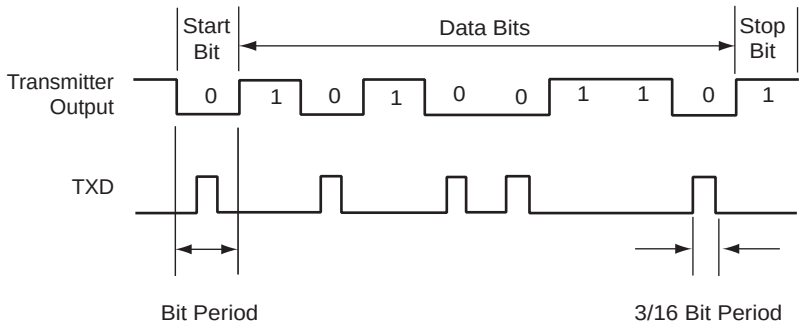
For baud rates up to and including 115.2 kbit/s, the RZI modulation scheme is used. “0” is represented by a light pulse of 3/16th of a bit time. Some examples of signal pulse duration are shown in Table 45-11.

Table 45-11. IrDA Pulse Duration

Baud Rate	Pulse Duration (3/16)
2.4 kbit/s	78.13 µs
9.6 kbit/s	19.53 µs
19.2 kbit/s	9.77 µs
38.4 kbit/s	4.88 µs
57.6 kbit/s	3.26 µs
115.2 kbit/s	1.63 µs

Figure 45-33 shows an example of character transmission.

Figure 45-33. IrDA Modulation



45.6.5.2 IrDA Baud Rate

Table 45-12 gives some examples of CD values, baud rate error and pulse duration. Note that the requirement on the maximum acceptable error of ±1.87% must be met.

Table 45-12. IrDA Baud Rate Error

Peripheral Clock	Baud Rate (bit/s)	CD	Baud Rate Error	Pulse Time (µs)
3,686,400	115,200	2	0.00%	1.63
20,000,000	115,200	11	1.38%	1.63
32,768,000	115,200	18	1.25%	1.63
40,000,000	115,200	22	1.38%	1.63
3,686,400	57,600	4	0.00%	3.26
20,000,000	57,600	22	1.38%	3.26
32,768,000	57,600	36	1.25%	3.26
40,000,000	57,600	43	0.93%	3.26
3,686,400	38,400	6	0.00%	4.88
20,000,000	38,400	33	1.38%	4.88
32,768,000	38,400	53	0.63%	4.88
40,000,000	38,400	65	0.16%	4.88
3,686,400	19,200	12	0.00%	9.77

Downloaded from Arrow.com.

Table 45-12. IrDA Baud Rate Error (Continued)

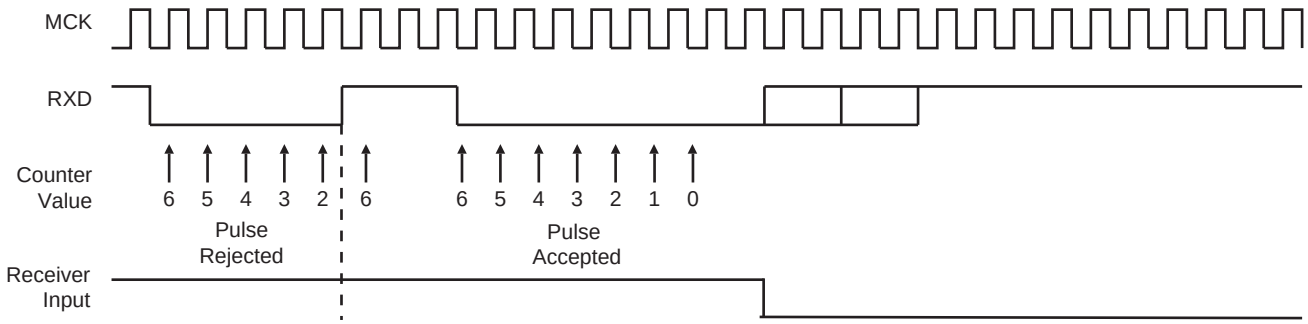
Peripheral Clock	Baud Rate (bit/s)	CD	Baud Rate Error	Pulse Time (µs)
20,000,000	19,200	65	0.16%	9.77
32,768,000	19,200	107	0.31%	9.77
40,000,000	19,200	130	0.16%	9.77
3,686,400	9,600	24	0.00%	19.53
20,000,000	9,600	130	0.16%	19.53
32,768,000	9,600	213	0.16%	19.53
40,000,000	9,600	260	0.16%	19.53
3,686,400	2,400	96	0.00%	78.13
20,000,000	2,400	521	0.03%	78.13
32,768,000	2,400	853	0.04%	78.13

45.6.5.3 IrDA Demodulator

The demodulator is based on the IrDA Receive filter comprised of an 8-bit down counter which is loaded with the value programmed in US_IF. When a falling edge is detected on the RXD pin, the Filter Counter starts counting down at the peripheral clock speed. If a rising edge is detected on the RXD pin, the counter stops and is reloaded with US_IF. If no rising edge is detected when the counter reaches 0, the input of the receiver is driven low during one bit time.

Figure 45-34 illustrates the operations of the IrDA demodulator.

Figure 45-34. IrDA Demodulator Operations



The programmed value in the US_IF register must always meet the following criteria:

$$t_{peripheral\ clock} \times (IRDA_FILTER + 3) < 1.41\ \mu s$$

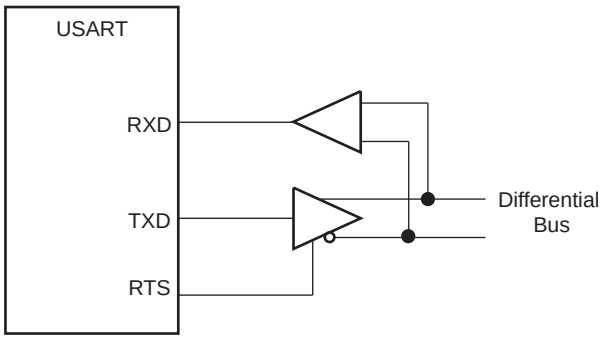
As the IrDA mode uses the same logic as the ISO7816, note that the FI_DI_RATIO field in US_FIDI must be set to a value higher than 0 in order to make sure IrDA communications operate correctly.

45.6.6 RS485 Mode

The USART features the RS485 mode to enable line driver control. While operating in RS485 mode, the USART behaves as though in Asynchronous or Synchronous mode and configuration of all the parameters is possible. The difference is that the RTS pin is driven high when the transmitter is operating. The behavior of the RTS pin is controlled by the TXEMPTY bit. A typical connection of the USART to an RS485 bus is shown in Figure 45-35.

Downloaded from Arrow.com.

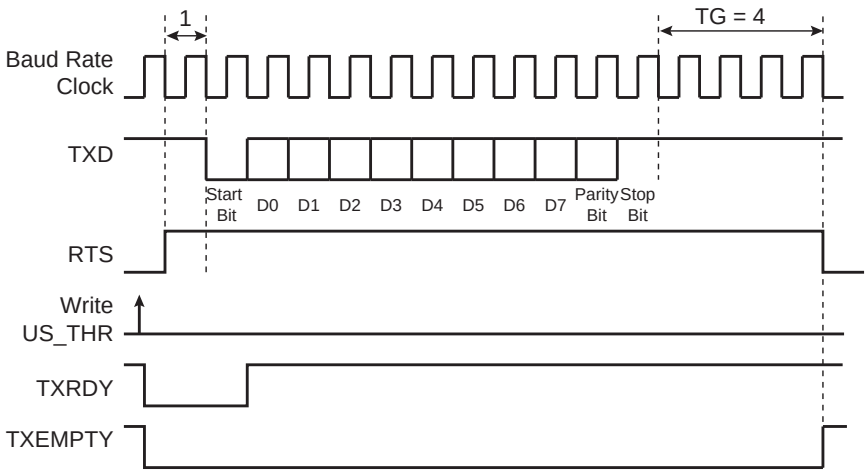
Figure 45-35. Typical Connection to a RS485 Bus



The USART is set in RS485 mode by writing the value 0x1 to the USART_MODE field in US_MR.

The RTS pin is at a level inverse to the TXEMPTY bit. Significantly, the RTS pin remains high when a timeguard is programmed so that the line can remain driven after the last character completion. Figure 45-36 gives an example of the RTS waveform during a character transmission when the timeguard is enabled.

Figure 45-36. Example of RTS Drive with Timeguard



45.6.7 Modem Mode

The USART features Modem mode, which enables control of the signals: DTR (Data Terminal Ready), DSR (Data Set Ready), RTS (Request to Send), CTS (Clear to Send), DCD (Data Carrier Detect) and RI (Ring Indicator). While operating in Modem mode, the USART behaves as a DTE (Data Terminal Equipment) as it drives DTR and RTS and can detect level change on DSR, DCD, CTS and RI.

Setting the USART in Modem mode is performed by writing the USART_MODE field in US_MR to the value 0x3. While operating in Modem mode, the USART behaves as though in Asynchronous mode and all the parameter configurations are available.

Table 45-13 gives the correspondence of the USART signals with modem connection standards.

Table 45-13. Circuit References

USART Pin	V24	CCITT	Direction
TXD	2	103	From terminal to modem
RTS	4	105	From terminal to modem
DTR	20	108.2	From terminal to modem

Downloaded from Arrow.com.

Table 45-13. Circuit References

USART Pin	V24	CCITT	Direction
RXD	3	104	From modem to terminal
CTS	5	106	From terminal to modem
DSR	6	107	From terminal to modem
DCD	8	109	From terminal to modem
RI	22	125	From terminal to modem

The control of the DTR output pin is performed by writing a 1 to the DTRDIS and DTREN bits respectively in US_CR. The disable command forces the corresponding pin to its inactive level, i.e., high. The enable command forces the corresponding pin to its active level, i.e., low. The RTS output pin is automatically controlled in this mode.

The level changes are detected on the RI, DSR, DCD and CTS pins. If an input change is detected, the RIIC, DSRIC, DCDIC and CTSIC bits in US_CSR are set respectively and can trigger an interrupt. The status is automatically cleared when US_CSR is read. Furthermore, the CTS automatically disables the transmitter when it is detected at its inactive state. If a character is being transmitted when the CTS rises, the character transmission is completed before the transmitter is actually disabled.

45.6.8 SPI Mode

The Serial Peripheral Interface (SPI) mode is a synchronous serial data link that provides communication with external devices in Master or Slave mode. It also enables communication between processors if an external processor is connected to the system.

The Serial Peripheral Interface is essentially a shift register that serially transmits data bits to other SPIs. During a data transfer, one SPI system acts as the “master” which controls the data flow, while the other devices act as “slaves" which have data shifted into and out by the master. Different CPUs can take turns being masters and one master may simultaneously shift data into multiple slaves. (Multiple master protocol is the opposite of single master protocol, where one CPU is always the master while all of the others are always slaves.) However, only one slave may drive its output to write data back to the master at any given time.

A slave device is selected when its NSS signal is asserted by the master. The USART in SPI Master mode can address only one SPI slave because it can generate only one NSS signal.

The SPI system consists of two data lines and two control lines:

- Master Out Slave In (MOSI): This data line supplies the output data from the master shifted into the input of the slave.
- Master In Slave Out (MISO): This data line supplies the output data from a slave to the input of the master.
- Serial Clock (SCK): This control line is driven by the master and regulates the flow of the data bits. The master may transmit data at a variety of baud rates. The SCK line cycles once for each bit that is transmitted.
- Slave Select (NSS): This control line allows the master to select or deselect the slave.

45.6.8.1 Modes of Operation

The USART can operate in SPI Master mode or in SPI Slave mode.

Operation in SPI Master mode is programmed by writing 0xE to the USART_MODE field in US_MR. In this case the SPI lines must be connected as described below:

- The MOSI line is driven by the output pin TXD
- The MISO line drives the input pin RXD
- The SCK line is driven by the output pin SCK

Downloaded from Arrow.com.

- The NSS line is driven by the output pin RTS

Operation in SPI Slave mode is programmed by writing to 0xF the USART_MODE field in US_MR. In this case the SPI lines must be connected as described below:

- The MOSI line drives the input pin RXD
- The MISO line is driven by the output pin TXD
- The SCK line drives the input pin SCK
- The NSS line drives the input pin CTS

In order to avoid unpredictable behavior, any change of the SPI mode must be followed by a software reset of the transmitter and of the receiver (except the initial configuration after a hardware reset). (See Section 45.6.8.4).

45.6.8.2 Baud Rate

In SPI mode, the baud rate generator operates in the same way as in USART Synchronous mode. See Section 45.6.1.3 "Baud Rate in Synchronous Mode or SPI Mode". However, there are some restrictions:

In SPI Master mode:

- The external clock SCK must not be selected (USCLKS ≠ 0x3), and the bit CLKO must be set to 1 in the US_MR, in order to generate correctly the serial clock on the SCK pin.
- To obtain correct behavior of the receiver and the transmitter, the value programmed in CD must be superior or equal to 6.
- If the divided peripheral clock is selected, the value programmed in CD must be even to ensure a 50:50 mark/space ratio on the SCK pin, this value can be odd if the peripheral clock is selected.

In SPI Slave mode:

- The external clock (SCK) selection is forced regardless of the value of the USCLKS field in the US_MR. Likewise, the value written in US_BRGR has no effect, because the clock is provided directly by the signal on the USART SCK pin.
- To obtain correct behavior of the receiver and the transmitter, the external clock (SCK) frequency must be at least 6 times lower than the system clock.

45.6.8.3 Data Transfer

Up to nine data bits are successively shifted out on the TXD pin at each rising or falling edge (depending of CPOL and CPHA) of the programmed serial clock. There is no Start bit, no Parity bit and no Stop bit.

The number of data bits is selected by the CHRL field and the MODE 9 bit in the US_MR. The nine bits are selected by setting the MODE 9 bit regardless of the CHRL field. The MSB data bit is always sent first in SPI mode (Master or Slave).

Four combinations of polarity and phase are available for data transfers. The clock polarity is programmed with the CPOL bit in the US_MR. The clock phase is programmed with the CPHA bit. These two parameters determine the edges of the clock signal upon which data is driven and sampled. Each of the two parameters has two possible states, resulting in four possible combinations that are incompatible with one another. Thus, a master/slave pair must use the same parameter pair values to communicate. If multiple slaves are used and fixed in different configurations, the master must reconfigure itself each time it needs to communicate with a different slave.

Table 45-14. SPI Bus Protocol Mode

SPI Bus Protocol Mode	CPOL	CPHA
0	0	1
1	0	0
2	1	1
3	1	0

Downloaded from Arrow.com.

Figure 45-37. SPI Transfer Format (CPHA = 1, 8 bits per transfer)

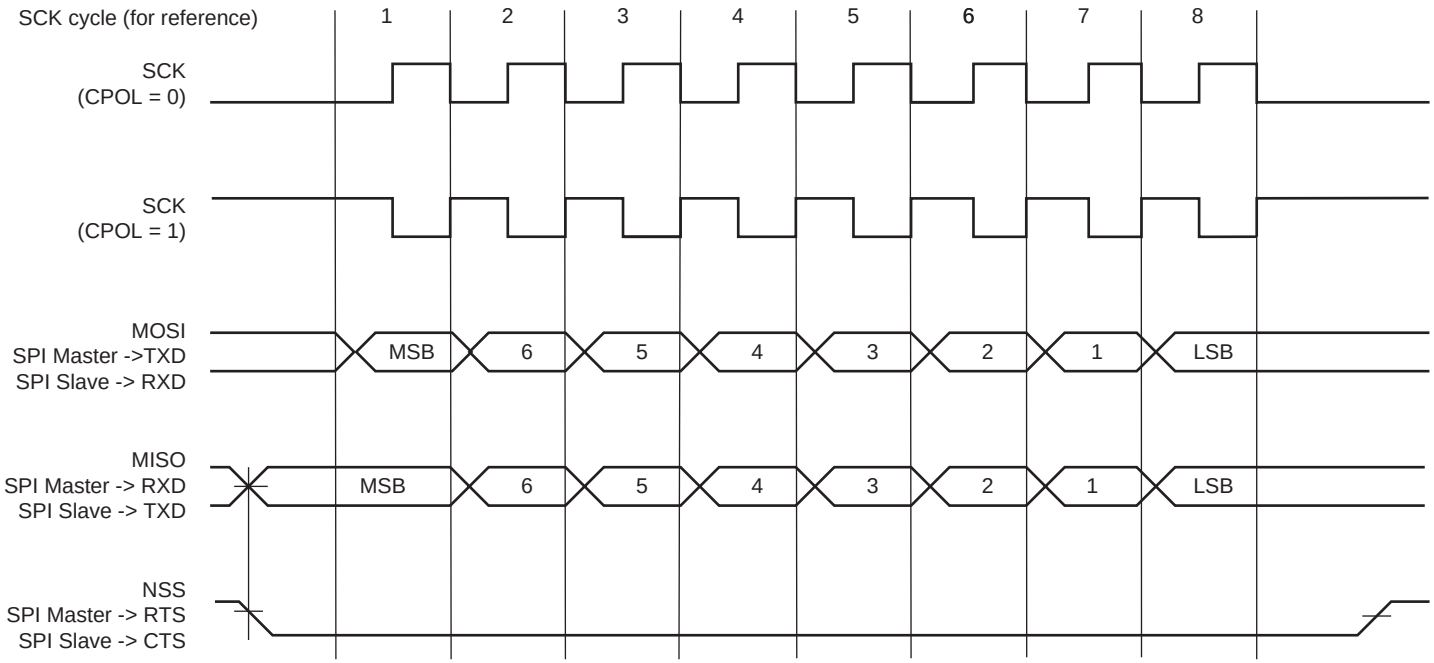
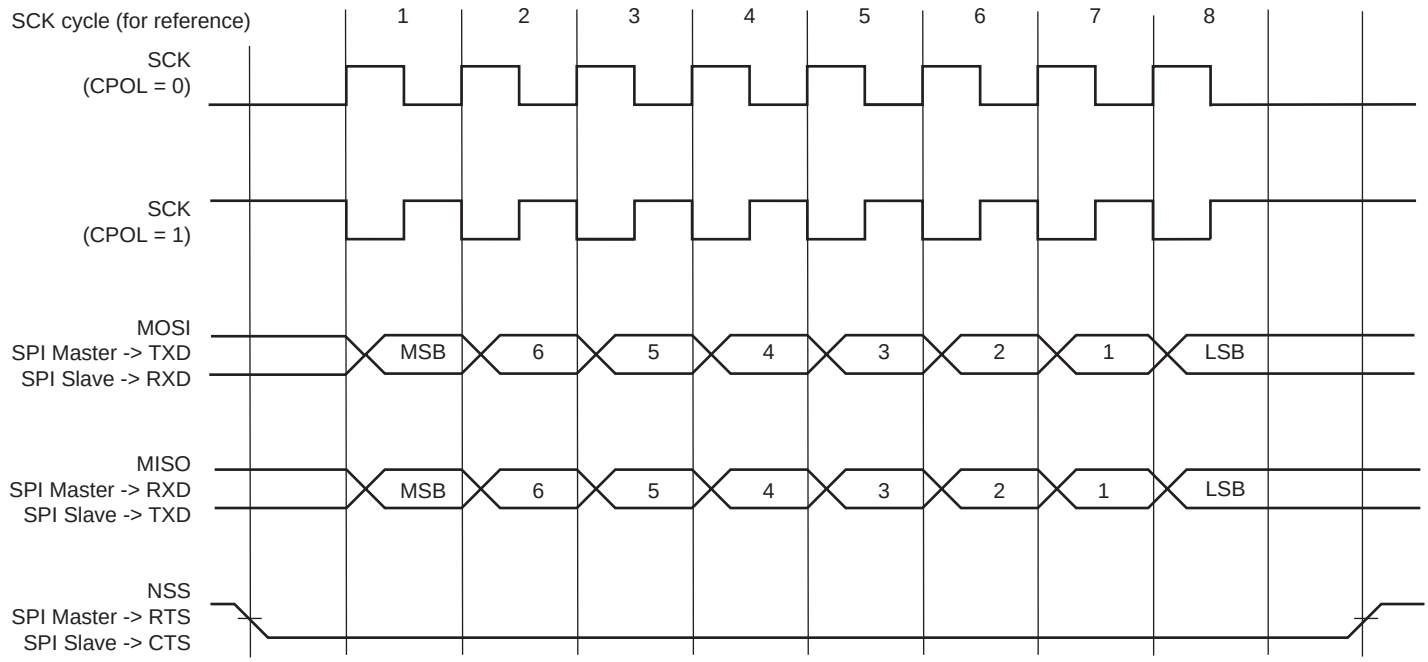


Figure 45-38. SPI Transfer Format (CPHA = 0, 8 bits per transfer)



45.6.8.4 Receiver and Transmitter Control

See Section 45.6.2 "Receiver and Transmitter Control"

45.6.8.5 Character Transmission

The characters are sent by writing in the Transmit Holding register (US_THR). An additional condition for transmitting a character can be added when the USART is configured in SPI Master mode. In the USART Mode Register (SPI_MODE) (USART_MR), the value configured on the bit WRDBT can prevent any character

Downloaded from Arrow.com.

transmission (even if US_THR has been written) while the receiver side is not ready (character not read). When WRDBT equals 0, the character is transmitted whatever the receiver status. If WRDBT is set to 1, the transmitter waits for the Receive Holding register (US_RHR) to be read before transmitting the character (RXRDY flag cleared), thus preventing any overflow (character loss) on the receiver side.

The chip select line is de-asserted for a period equivalent to three bits between the transmission of two data.

The transmitter reports two status bits in US_CSR: TXRDY (Transmitter Ready), which indicates that US_THR is empty and TXEMPTY, which indicates that all the characters written in US_THR have been processed. When the current character processing is completed, the last character written in US_THR is transferred into the Shift register of the transmitter and US_THR becomes empty, thus TXRDY rises.

Both TXRDY and TXEMPTY bits are low when the transmitter is disabled. Writing a character in US_THR while TXRDY is low has no effect and the written character is lost.

If the USART is in SPI Slave mode and if a character must be sent while the US_THR is empty, the UNRE (Underrun Error) bit is set. The TXD transmission line stays at high level during all this time. The UNRE bit is cleared by writing a 1 to the RSTSTA (Reset Status) bit in US_CR.

In SPI Master mode, the slave select line (NSS) is asserted at low level one t_{bit} (t_{bit} being the nominal time required to transmit a bit) before the transmission of the MSB bit and released at high level one t_{bit} after the transmission of the LSB bit. So, the slave select line (NSS) is always released between each character transmission and a minimum delay of three t_{bit} always inserted. However, in order to address slave devices supporting the CSAAT mode (Chip Select Active After Transfer), the slave select line (NSS) can be forced at low level by writing a 1 to the RCS bit in the US_CR. The slave select line (NSS) can be released at high level only by writing a 1 to the FCS bit in the US_CR (for example, when all data have been transferred to the slave device).

In SPI Slave mode, the transmitter does not require a falling edge of the slave select line (NSS) to initiate a character transmission but only a low level. However, this low level must be present on the slave select line (NSS) at least one t_{bit} before the first serial clock cycle corresponding to the MSB bit.

45.6.8.6 Character Reception

When a character reception is completed, it is transferred to the Receive Holding register (US_RHR) and the RXRDY bit in the Status register (US_CSR) rises. If a character is completed while RXRDY is set, the OVRE (Overrun Error) bit is set. The last character is transferred into US_RHR and overwrites the previous one. The OVRE bit is cleared by writing a 1 to the RSTSTA (Reset Status) bit in the US_CR.

To ensure correct behavior of the receiver in SPI Slave mode, the master device sending the frame must ensure a minimum delay of one t_{bit} between each character transmission. The receiver does not require a falling edge of the slave select line (NSS) to initiate a character reception but only a low level. However, this low level must be present on the slave select line (NSS) at least one t_{bit} before the first serial clock cycle corresponding to the MSB bit.

45.6.8.7 Receiver Timeout

Because the receiver baud rate clock is active only during data transfers in SPI mode, a receiver timeout is impossible in this mode, whatever the time-out value is (field TO) in the US_RTOR.

45.6.9 LIN Mode

The LIN mode provides master node and slave node connectivity on a LIN bus.

The LIN (Local Interconnect Network) is a serial communication protocol which efficiently supports the control of mechatronic nodes in distributed automotive applications.

The main properties of the LIN bus are:

- Single master/multiple slaves concept
- Low-cost silicon implementation based on common UART/SCI interface hardware, an equivalent in software, or as a pure state machine.

Downloaded from Arrow.com.

- Self synchronization without quartz or ceramic resonator in the slave nodes
- Deterministic signal transmission
- Low cost single-wire implementation
- Speed up to 20 kbit/s

LIN provides cost efficient bus communication where the bandwidth and versatility of CAN are not required.

The LIN mode enables processing LIN frames with a minimum of action from the microprocessor.

45.6.9.1 Modes of Operation

The USART can act either as a LIN master node or as a LIN slave node.

The node configuration is chosen by setting the USART_MODE field in the USART Mode register (US_MR):

- LIN master node (USART_MODE = 0xA)
- LIN slave node (USART_MODE = 0xB)

In order to avoid unpredictable behavior, any change of the LIN node configuration must be followed by a software reset of the transmitter and of the receiver (except the initial node configuration after a hardware reset). (See Section 45.6.9.3 "Receiver and Transmitter Control".)

45.6.9.2 Baud Rate Configuration

See Section 45.6.1.1 "Baud Rate in Asynchronous Mode"

- LIN master node: The baud rate is configured in US_BRGR.
- LIN slave node: The initial baud rate is configured in US_BRGR. This configuration is automatically copied in the LIN Baud Rate register (US_LINBRR) when writing US_BRGR. After the synchronization procedure, the baud rate is updated in US_LINBRR.

45.6.9.3 Receiver and Transmitter Control

See Section 45.6.2 "Receiver and Transmitter Control"

45.6.9.4 Character Transmission

See Section 45.6.3.1 "Transmitter Operations".

45.6.9.5 Character Reception

See Section 45.6.3.7 "Receiver Operations".

45.6.9.6 Header Transmission (Master Node Configuration)

All the LIN frames start with a header which is sent by the master node and consists of a Synch Break Field, Synch Field and Identifier Field.

So in master node configuration, the frame handling starts with the sending of the header.

The header is transmitted as soon as the identifier is written in the LIN Identifier register (US_LINIR). At this moment the flag TXRDY falls.

The Break Field, the Synch Field and the Identifier Field are sent automatically one after the other.

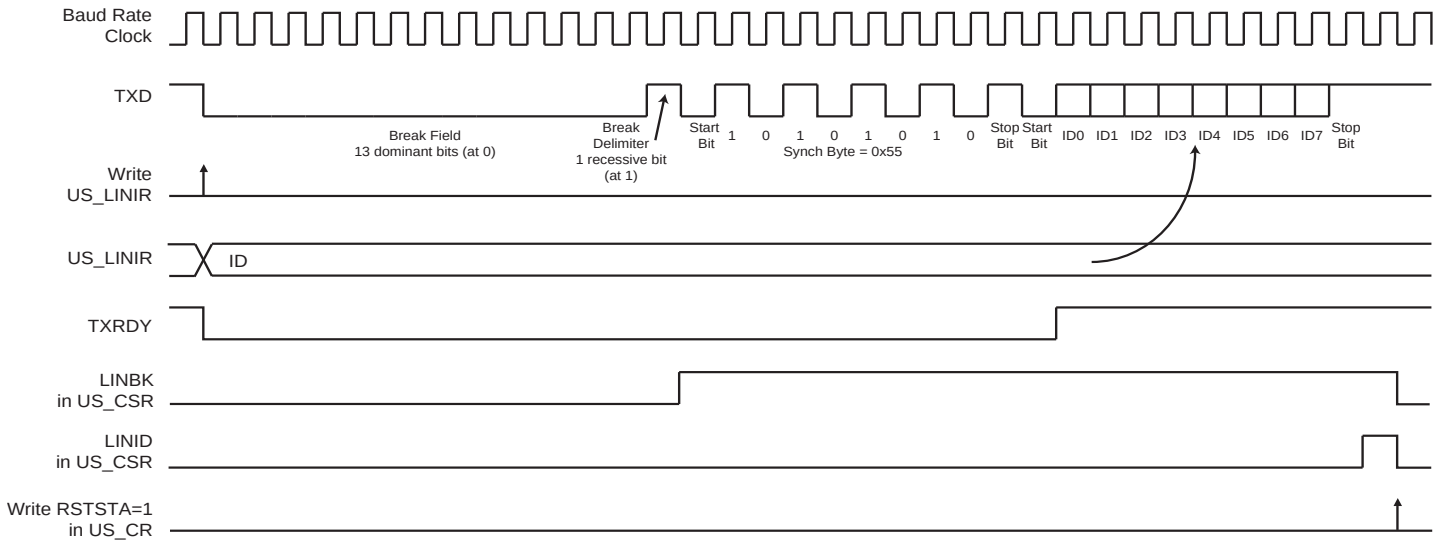
The Break Field consists of 13 dominant bits and 1 recessive bit, the Synch Field is the character 0x55 and the Identifier corresponds to the character written in the LIN Identifier register (US_LINIR). The Identifier parity bits can be automatically computed and sent (see Section 45.6.9.9 "Identifier Parity").

The flag TXRDY rises when the identifier character is transferred into the Shift register of the transmitter.

As soon as the Synch Break Field is transmitted, the flag LINBK in US_CSR is set to 1. Likewise, as soon as the Identifier Field is sent, the flag bit LINID in the US_CSR is set to 1. These flags are reset by writing a 1 to the bit RSTSTA in US_CR.

Downloaded from Arrow.com.

Figure 45-39. Header Transmission



45.6.9.7 Header Reception (Slave Node Configuration)

All the LIN frames start with a header which is sent by the master node and consists of a Synch Break Field, Synch Field and Identifier Field.

In slave node configuration, the frame handling starts with the reception of the header.

The USART uses a break detection threshold of 11 nominal bit times at the actual baud rate. At any time, if 11 consecutive recessive bits are detected on the bus, the USART detects a Break Field. As long as a Break Field has not been detected, the USART stays idle and the received data are not taken in account.

When a Break Field has been detected, the flag LINBK in US_CSR is set to 1 and the USART expects the Synch Field character to be 0x55. This field is used to update the actual baud rate in order to stay synchronized (see Section 45.6.9.8 ”Slave Node Synchronization”). If the received Synch character is not 0x55, an Inconsistent Synch Field error is generated (see Section 45.6.9.14 ”LIN Errors”).

After receiving the Synch Field, the USART expects to receive the Identifier Field.

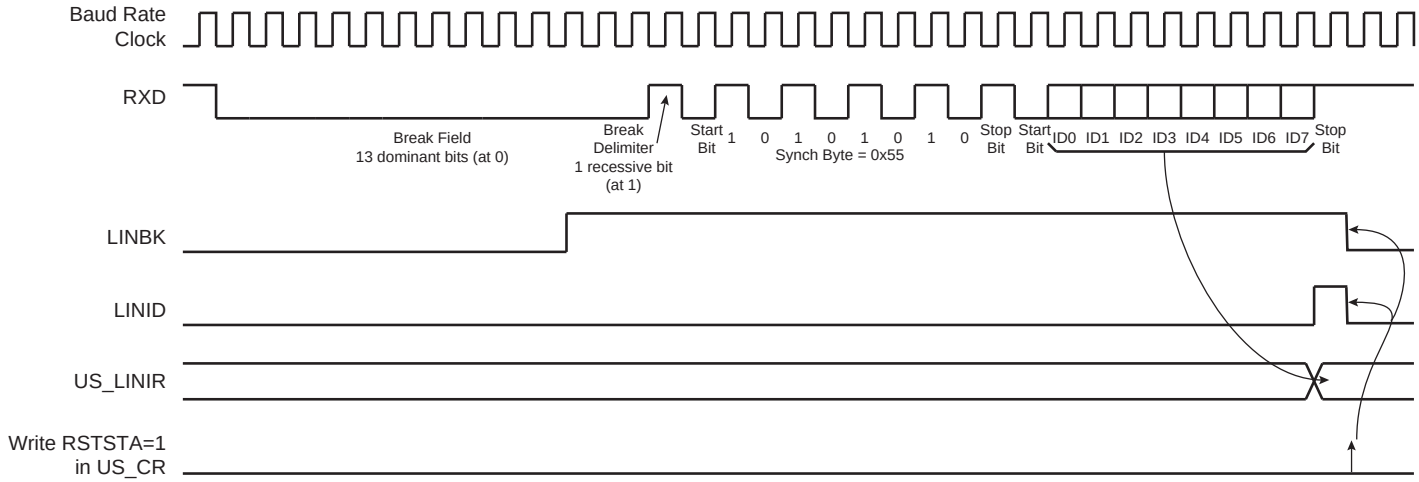
When the Identifier Field has been received, the flag bit LINID in the US_CSR is set to 1. At this moment the field IDCHR in the LIN Identifier register (US_LINIR) is updated with the received character. The Identifier parity bits can be automatically computed and checked (see Section 45.6.9.9 ”Identifier Parity”).

If the Header is not entirely received within the time given by the maximum length of the header $t_{Header_Maximum}$, the error flag LINHTE in the US_CSR is set to 1.

The flag bits LINID, LINBK and LINHTE are reset by writing a 1 to the bit RSTSTA in US_CR.

Downloaded from Arrow.com.

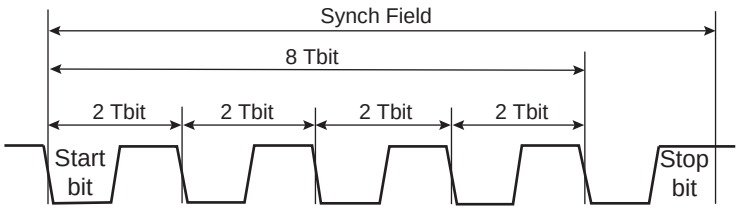
Figure 45-40. Header Reception



45.6.9.8 Slave Node Synchronization

The synchronization is done only in slave node configuration. The procedure is based on time measurement between falling edges of the Synch Field. The falling edges are available in distances of 2, 4, 6 and 8 bit times.

Figure 45-41. Synch Field



The time measurement is made by a 19-bit counter clocked by the sampling clock (see Section 45.6.1 "Baud Rate Generator").

When the start bit of the Synch Field is detected, the counter is reset. Then during the next eight t_{bit} of the Synch Field, the counter is incremented. At the end of these eight t_{bit}, the counter is stopped. At this moment, the 16 most significant bits of the counter (value divided by 8) give the new clock divider (LINCD) and the three least significant bits of this value (the remainder) give the new fractional part (LINFP).

Once the Synch Field has been entirely received, the clock divider (LINCD) and the fractional part (LINFP) are updated in the LIN Baud Rate register (US_LINBRR) with the computed values, if the Synchronization is not disabled by the SYNCDIS bit in the LIN Mode register (US_LINMR).

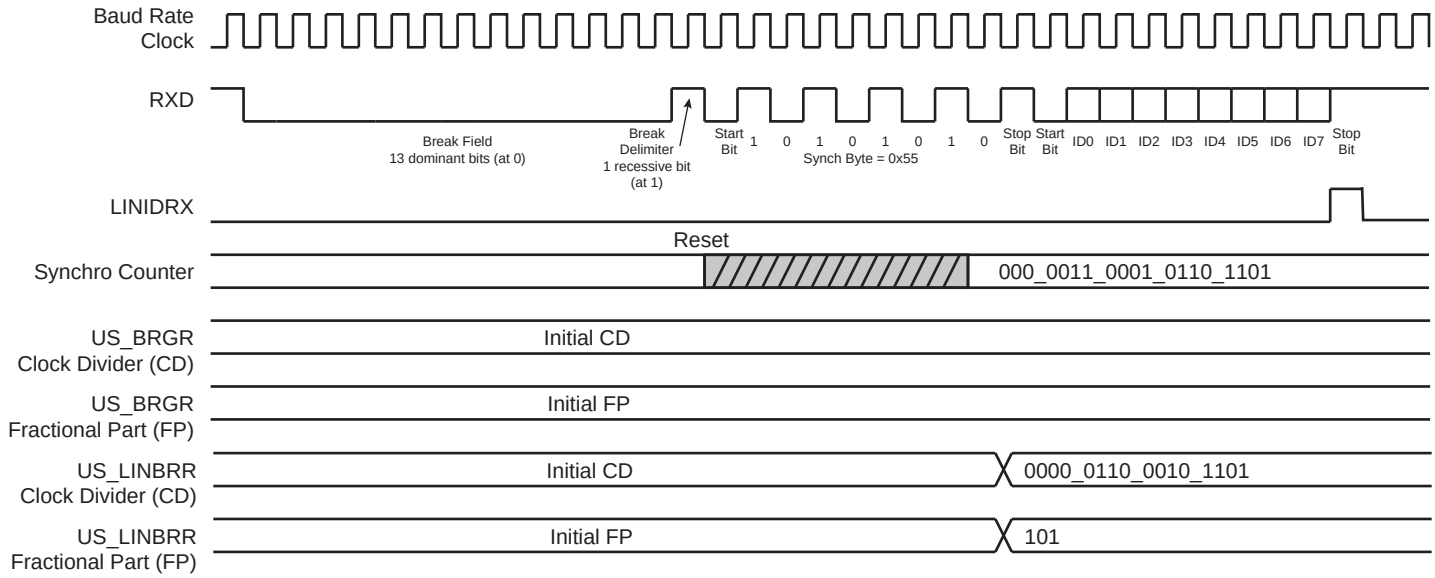
After reception of the Synch Field:

- If it appears that the computed baud rate deviation compared to the initial baud rate is superior to the maximum tolerance FTol_Unsynch (±15%), then the clock divider (LINCD) and the fractional part (LINFP) are not updated, and the error flag LINSTE in US_CSR is set to 1.
- If it appears that the sampled Synch character is not equal to 0x55, then the clock divider (LINCD) and the fractional part (LINFP) are not updated, and the error flag LINISFE in US_CSR is set to 1.

Flags LINSTE and LINISFE are reset by writing bit RSTSTA to 1 in US_CR.

Downloaded from Arrow.com.

Figure 45-42. Slave Node Synchronization



The accuracy of the synchronization depends on several parameters:

- Nominal clock frequency (f_{Nom}) (the theoretical slave node clock frequency)
- Baud Rate
- Oversampling (OVER = 0 => 16X or OVER = 1 => 8X)

The following formula is used to compute the deviation of the slave bit rate relative to the master bit rate after synchronization (f_{SLAVE} is the real slave node clock frequency):

$$Baudrate_deviation = \left(100 \times \frac{[\alpha \times 8 \times (2 - OVER) + \beta] \times Baudrate}{8 \times f_{SLAVE}}\right)\%$$

$$Baudrate_deviation = \left(100 \times \frac{[\alpha \times 8 \times (2 - OVER) + \beta] \times Baudrate}{8 \times \left(\frac{f_{TOL_UNSYNCH}}{100}\right) \times f_{Nom}}\right)\%$$

$$-0.5 \le \alpha \le +0.5 \quad -1 < \beta < +1$$

$f_{TOL_UNSYNCH}$ is the deviation of the real slave node clock from the nominal clock frequency. The LIN Standard imposes that it must not exceed ±15%. The LIN Standard imposes also that for communication between two nodes, their bit rate must not differ by more than ±2%. This means that the baudrate_deviation must not exceed ±1%.

It follows from that, a minimum value for the nominal clock frequency:

$$f_{Nom}(min) = \left(100 \times \frac{[0.5 \times 8 \times (2 - OVER) + 1] \times Baudrate}{8 \times \left(\frac{-15}{100} + 1\right) \times 1\%}\right) Hz$$

Examples:

Downloaded from Arrow.com.

- Baud rate = 20 kbit/s, OVER = 0 (Oversampling 16X) => f_{Nom}(min) = 2.64 MHz
- Baud rate = 20 kbit/s, OVER = 1 (Oversampling 8X) => f_{Nom}(min) = 1.47 MHz
- Baud rate = 1 kbit/s, OVER = 0 (Oversampling 16X) => f_{Nom}(min) = 132 kHz
- Baud rate = 1 kbit/s, OVER = 1 (Oversampling 8X) => f_{Nom}(min) = 74 kHz

45.6.9.9 Identifier Parity

A protected identifier consists of two subfields: the identifier and the identifier parity. Bits 0 to 5 are assigned to the identifier and bits 6 and 7 are assigned to the parity.

The USART interface can generate/check these parity bits, but this feature can also be disabled. The user can choose between two modes by the PARDIS bit of US_LINMR:

- PARDIS = 0:
  - During header transmission, the parity bits are computed and sent with the six least significant bits of the IDCHR field of the LIN Identifier register (US_LINIR). The bits 6 and 7 of this register are discarded.
  - During header reception, the parity bits of the identifier are checked. If the parity bits are wrong, an Identifier Parity error occurs (see Section 45.6.3.8). Only the six least significant bits of the IDCHR field are updated with the received Identifier. The bits 6 and 7 are stuck to 0.
- PARDIS = 1:
  - During header transmission, all the bits of the IDCHR field of the LIN Identifier register (US_LINIR) are sent on the bus.
  - During header reception, all the bits of the IDCHR field are updated with the received Identifier.

45.6.9.10 Node Action

Depending on the identifier, the node is affected – or not – by the LIN response. Consequently, after sending or receiving the identifier, the USART must be configured. There are three possible configurations:

- PUBLISH: the node sends the response.
- SUBSCRIBE: the node receives the response.
- IGNORE: the node is not concerned by the response, it does not send and does not receive the response.

This configuration is made by the field Node Action (NACT) in the US_LINMR (see Section 45.7.32).

Example: a LIN cluster that contains a master and two slaves:

- Data transfer from the master to the slave1 and to the slave2:

  NACT(master)=PUBLISH

  NACT(slave1)=SUBSCRIBE

  NACT(slave2)=SUBSCRIBE
- Data transfer from the master to the slave1 only:

  NACT(master)=PUBLISH

  NACT(slave1)=SUBSCRIBE

  NACT(slave2)=IGNORE
- Data transfer from the slave1 to the master:

  NACT(master)=SUBSCRIBE

  NACT(slave1)=PUBLISH

  NACT(slave2)=IGNORE
- Data transfer from the slave1 to the slave2:

  NACT(master)=IGNORE

Downloaded from Arrow.com.

NACT(slave1)=PUBLISH

NACT(slave2)=SUBSCRIBE

- Data transfer from the slave2 to the master and to the slave1:

  NACT(master)=SUBSCRIBE

  NACT(slave1)=SUBSCRIBE

  NACT(slave2)=PUBLISH

45.6.9.11 Response Data Length

The LIN response data length is the number of data fields (bytes) of the response excluding the checksum.

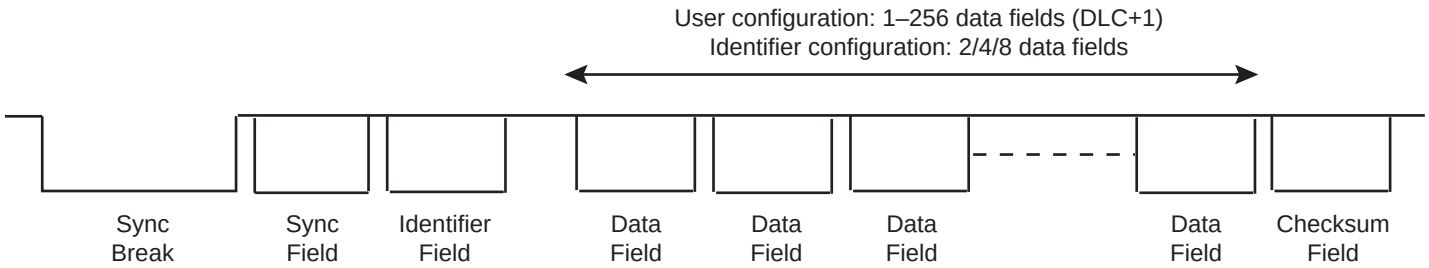
The response data length can either be configured by the user or be defined automatically by bits 4 and 5 of the Identifier (compatibility to LIN Specification 1.1). The user can choose between these two modes by the DLM bit of US_LINMR:

- DLM = 0: The response data length is configured by the user via the DLC field of US_LINMR. The response data length is equal to (DLC + 1) bytes. DLC can be programmed from 0 to 255, so the response can contain from 1 data byte up to 256 data bytes.
- DLM = 1: The response data length is defined by the Identifier (IDCHR in US_LINIR) according to the table below. The DLC field of US_LINMR is discarded. The response can contain 2 or 4 or 8 data bytes.

Table 45-15. Response Data Length if DLM = 1

IDCHR[5]	IDCHR[4]	Response Data Length [Bytes]
0	0	2
0	1	2
1	0	4
1	1	8

Figure 45-43. Response Data Length



45.6.9.12 Checksum

The last field of a frame is the checksum. The checksum contains the inverted 8-bit sum with carry, over all data bytes or all data bytes and the protected identifier. Checksum calculation over the data bytes only is called classic checksum and it is used for communication with LIN 1.3 slaves. Checksum calculation over the data bytes and the protected identifier byte is called enhanced checksum and it is used for communication with LIN 2.0 slaves.

The USART can be configured to:

- Send/Check an Enhanced checksum automatically (CHKDIS = 0 & CHKTYP = 0)
- Send/Check a Classic checksum automatically (CHKDIS = 0 & CHKTYP = 1)
- Not send/check a checksum (CHKDIS = 1)

This configuration is made by the Checksum Type (CHKTYP) and Checksum Disable (CHKDIS) fields of US_LINMR.

Downloaded from Arrow.com.

If the checksum feature is disabled, the user can send it manually all the same, by considering the checksum as a normal data byte and by adding 1 to the response data length (see Section 45.6.9.11).

45.6.9.13 Frame Slot Mode

This mode is useful only for master nodes. It complies with the following rule: each frame slot should be longer than or equal to $t_{Frame_Maximum}$.

If the Frame Slot mode is enabled (FSDIS = 0) and a frame transfer has been completed, the TXRDY flag is set again only after $t_{Frame_Maximum}$ delay, from the start of frame. So the master node cannot send a new header if the frame slot duration of the previous frame is inferior to $t_{Frame_Maximum}$.

If the Frame Slot mode is disabled (FSDIS = 1) and a frame transfer has been completed, the TXRDY flag is set again immediately.

The $t_{Frame_Maximum}$ is calculated as below:

If the Checksum is sent (CHKDIS = 0):

$t_{Header_Nominal} = 34 \times t_{bit}$

$t_{Response_Nominal} = 10 \times (NData + 1) \times t_{bit}$

$t_{Frame_Maximum} = 1.4 \times (t_{Header_Nominal} + t_{Response_Nominal} + 1)$(1)

$t_{Frame_Maximum} = 1.4 \times (34 + 10 \times (DLC + 1 + 1) + 1) \times t_{bit}$

$t_{Frame_Maximum} = (77 + 14 \times DLC) \times t_{bit}$

If the Checksum is not sent (CHKDIS = 1):

$t_{Header_Nominal} = 34 \times t_{bit}$

$t_{Response_Nominal} = 10 \times NData \times t_{bit}$

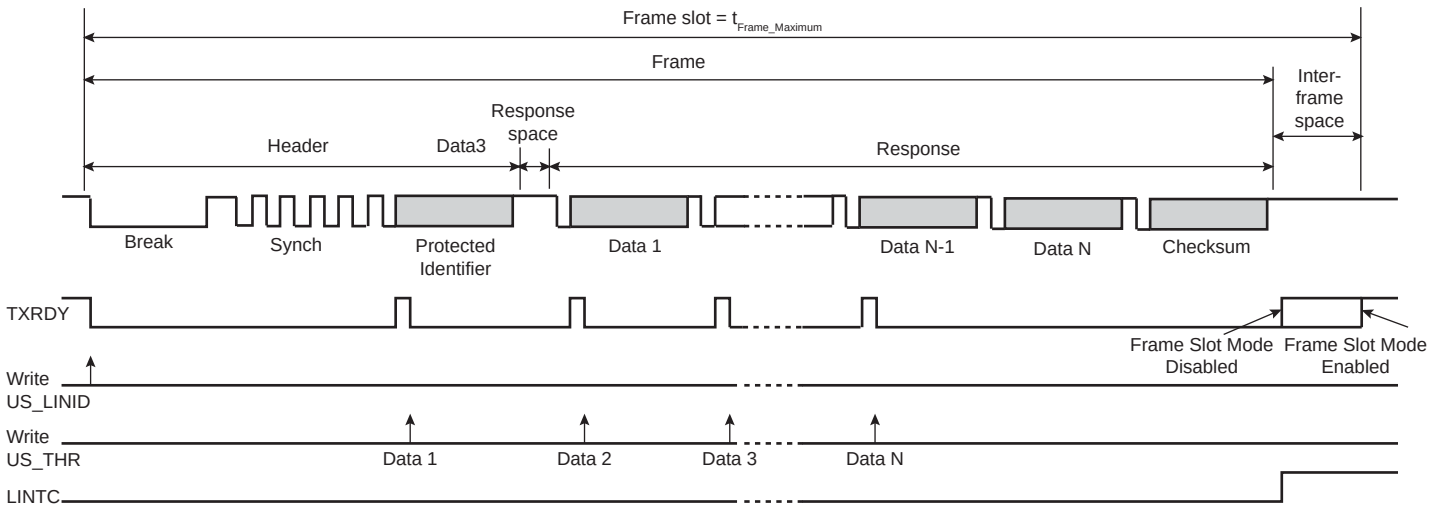
$t_{Frame_Maximum} = 1.4 \times (t_{Header_Nominal} + t_{Response_Nominal} + 1)$(1)

$t_{Frame_Maximum} = 1.4 \times (34 + 10 \times (DLC + 1) + 1) \times t_{bit}$

$t_{Frame_Maximum} = (63 + 14 \times DLC) \times t_{bit}$

Note: 1. The term “+1” leads to an integer result for $t_{Frame_Maximum}$ (LIN Specification 1.3).

Figure 45-44. Frame Slot Mode



Downloaded from Arrow.com.

45.6.9.14 LIN Errors

Bit Error

This error is generated in master of slave node configuration, when the USART is transmitting and if the transmitted value on the Tx line is different from the value sampled on the Rx line. If a bit error is detected, the transmission is aborted at the next byte border.

This error is reported by flag LINBE in US_CSR.

Inconsistent Synch Field Error

This error is generated in slave node configuration, if the Synch Field character received is other than 0x55.

This error is reported by flag LINISFE in the US_CSR.

Identifier Parity Error

This error is generated in slave node configuration, if the parity of the identifier is wrong. This error can be generated only if the parity feature is enabled (PARDIS = 0).

This error is reported by flag LINIPE in the US_CSR.

Checksum Error

This error is generated in master of slave node configuration, if the received checksum is wrong. This flag can be set to 1 only if the checksum feature is enabled (CHKDIS = 0).

This error is reported by flag LINCE in the US_CSR.

Slave Not Responding Error

This error is generated in master of slave node configuration, when the USART expects a response from another node (NACT = SUBSCRIBE) but no valid message appears on the bus within the time given by the maximum length of the message frame, $t_{Frame_Maximum}$ (see Section 45.6.9.13). This error is disabled if the USART does not expect any message (NACT = PUBLISH or NACT = IGNORE).

This error is reported by flag LINSNRE in the US_CSR.

Synch Tolerance Error

This error is generated in slave node configuration if, after the clock synchronization procedure, it appears that the computed baudrate deviation compared to the initial baudrate is superior to the maximum tolerance FTol_Unsynch (±15%).

This error is reported by flag LINSTE in the US_CSR.

Header Timeout Error

This error is generated in slave node configuration, if the Header is not entirely received within the time given by the maximum length of the Header, $t_{Header_Maximum}$.

This error is reported by flag LINHTE in the US_CSR.

45.6.9.15 LIN Frame Handling

Master Node Configuration

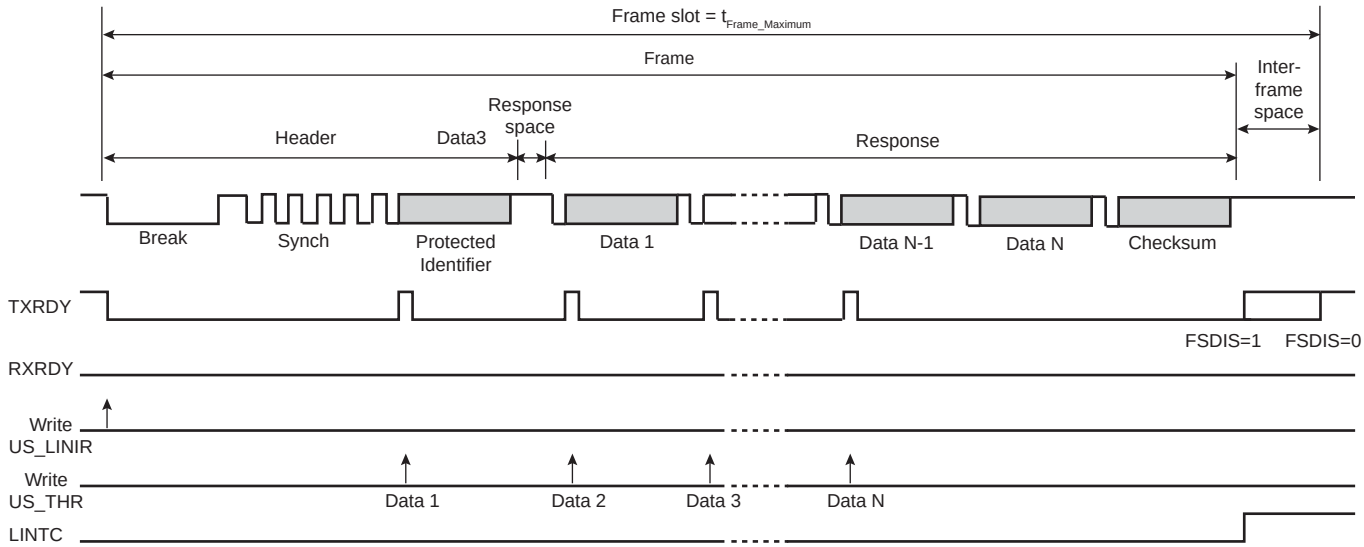
- Write TXEN and RXEN in US_CR to enable both the transmitter and the receiver.
- Write USART_MODE in US_MR to select the LIN mode and the master node configuration.
- Write CD and FP in US_BRGR to configure the baud rate.
- Write NACT, PARDIS, CHKDIS, CHKTYPE, DLCM, FSDIS and DLC in US_LINMR to configure the frame transfer.
- Check that TXRDY in US_CSR is set to 1
- Write IDCHR in US_LINIR to send the header

Downloaded from Arrow.com.

What comes next depends on the NACT configuration:

- Case 1: NACT = PUBLISH, the USART sends the response
  - Wait until TXRDY in US_CSR rises
  - Write TCHR in US_THR to send a byte
  - If all the data have not been written, redo the two previous steps
  - Wait until LINTC in US_CSR rises
  - Check the LIN errors
- Case 2: NACT = SUBSCRIBE, the USART receives the response
  - Wait until RXRDY in US_CSR rises
  - Read RCHR in US_RHR
  - If all the data have not been read, redo the two previous steps
  - Wait until LINTC in US_CSR rises
  - Check the LIN errors
- Case 3: NACT = IGNORE, the USART is not concerned by the response
  - Wait until LINTC in US_CSR rises
  - Check the LIN errors

Figure 45-45. Master Node Configuration, NACT = PUBLISH



Downloaded from Arrow.com.

Figure 45-46. Master Node Configuration, NACT = SUBSCRIBE

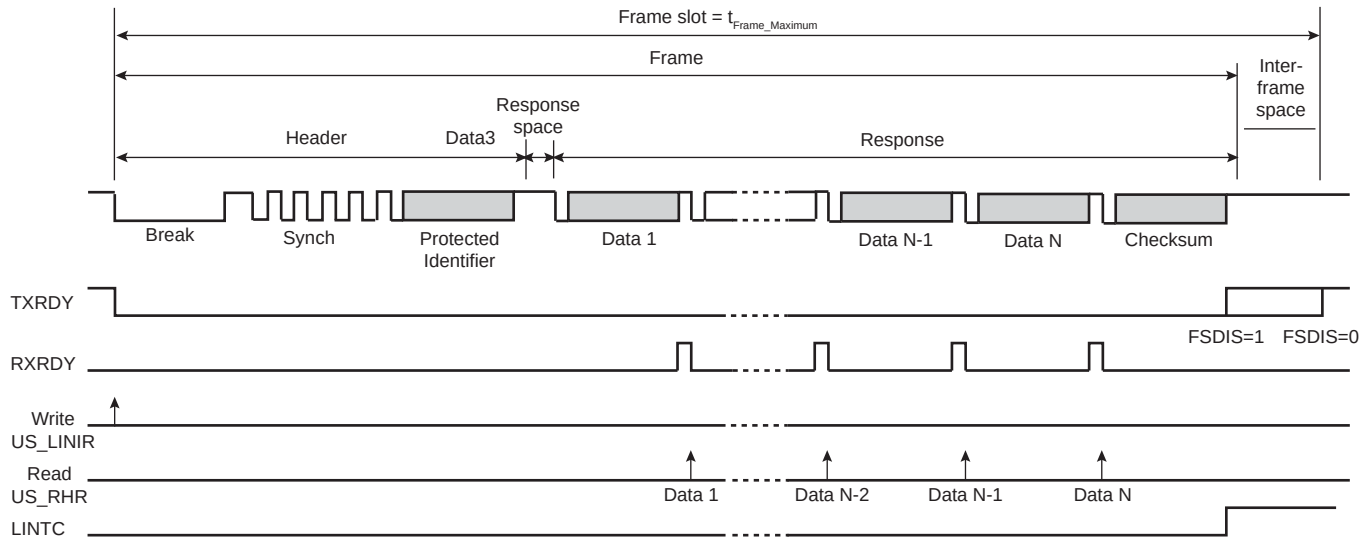
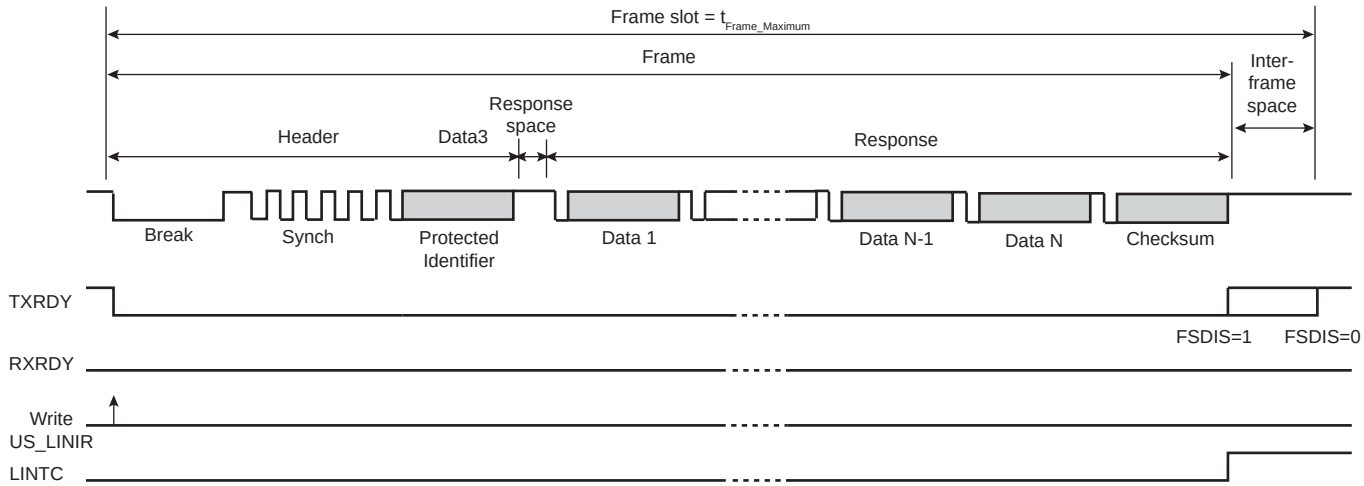


Figure 45-47. Master Node Configuration, NACT = IGNORE



Slave Node Configuration

- Write TXEN and RXEN in US_CR to enable both the transmitter and the receiver.
- Write USART_MODE in US_MR to select the LIN mode and the slave node configuration.
- Write CD and FP in US_BRGR to configure the baud rate.
- Wait until LINID in US_CSR rises
- Check LINISFE and LINPE errors
- Read IDCHR in US_RHR
- Write NACT, PARDIS, CHKDIS, CHKTYPE, DLCM and DLC in US_LINMR to configure the frame transfer.

IMPORTANT: If the NACT configuration for this frame is PUBLISH, the US_LINMR must be written with NACT = PUBLISH even if this field is already correctly configured, in order to set the TXREADY flag and the corresponding write transfer request.

What comes next depends on the NACT configuration:

- Case 1: NACT = PUBLISH, the LIN controller sends the response
  - Wait until TXRDY in US_CSR rises

Downloaded from Arrow.com.

- Write TCHR in US_THR to send a byte
- If all the data have not been written, redo the two previous steps
- Wait until LINTC in US_CSR rises
- Check the LIN errors

- Case 2: NACT = SUBSCRIBE, the USART receives the response
  - Wait until RXRDY in US_CSR rises
  - Read RCHR in US_RHR
  - If all the data have not been read, redo the two previous steps
  - Wait until LINTC in US_CSR rises
  - Check the LIN errors
- Case 3: NACT = IGNORE, the USART is not concerned by the response
  - Wait until LINTC in US_CSR rises
  - Check the LIN errors

Figure 45-48. Slave Node Configuration, NACT = PUBLISH

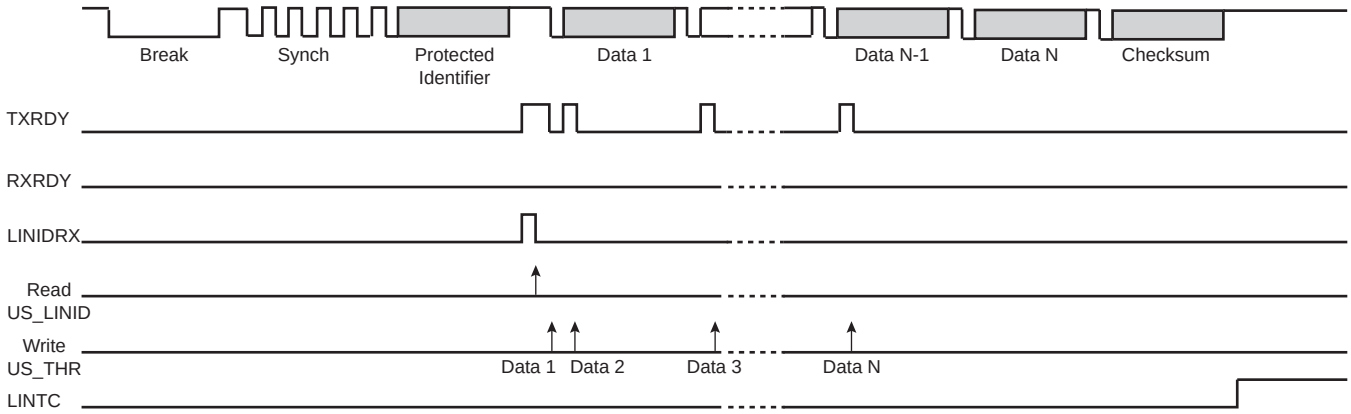
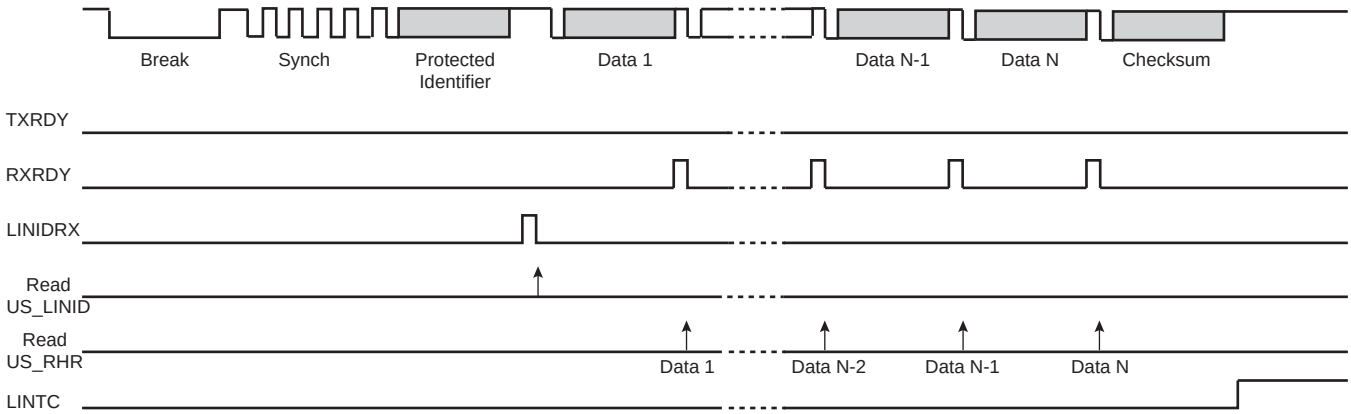
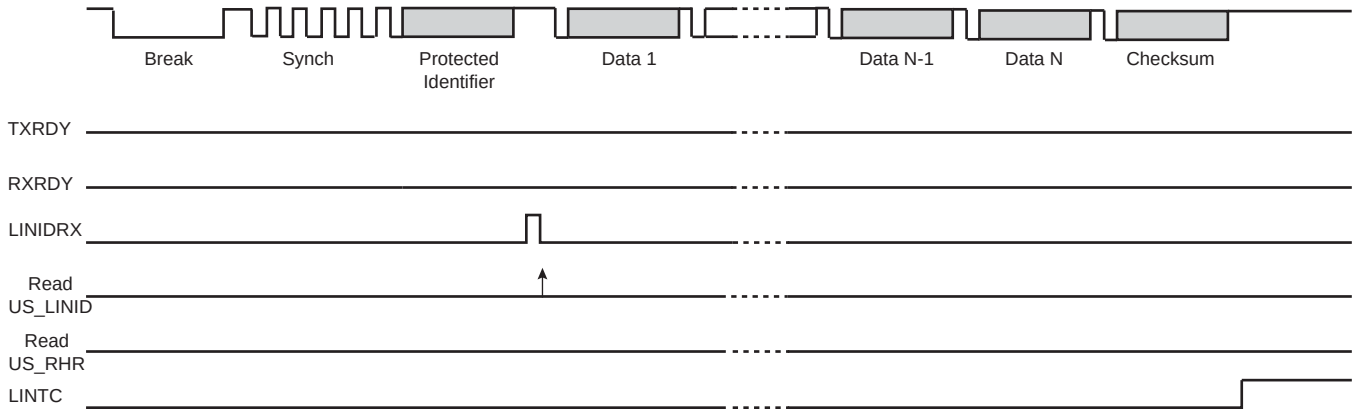


Figure 45-49. Slave Node Configuration, NACT = SUBSCRIBE



Downloaded from Arrow.com.

Figure 45-50. Slave Node Configuration, NACT = IGNORE



45.6.9.16 LIN Frame Handling with the DMAC

The USART can be used in association with the DMAC in order to transfer data directly into/from the on- and off-chip memories without any processor intervention.

The DMAC uses the trigger flags, TXRDY and RXRDY, to write or read into the USART. The DMAC always writes in the Transmit Holding register (US_THR) and it always reads in the Receive Holding register (US_RHR). The size of the data written or read by the DMAC in the USART is always a byte.

Master Node Configuration

The user can choose between two DMAC modes by the PDCM bit in the US_LINMR:

- PDCM = 1: the LIN configuration is stored in the WRITE buffer and it is written by the DMAC in the Transmit Holding register US_THR (instead of the LIN Mode register US_LINMR). Because the DMAC transfer size is limited to a byte, the transfer is split into two accesses. During the first access the bits, NACT, PARDIS, CHKDIS, CHKTYP, DLM and FSDIS are written. During the second access the 8-bit DLC field is written.
- PDCM = 0: the LIN configuration is not stored in the WRITE buffer and it must be written by the user in US_LINMR.

The WRITE buffer also contains the Identifier and the DATA, if the USART sends the response (NACT = PUBLISH).

The READ buffer contains the DATA if the USART receives the response (NACT = SUBSCRIBE).

Downloaded from Arrow.com.

Figure 45-51. Master Node with DMAC (PDCM = 1)

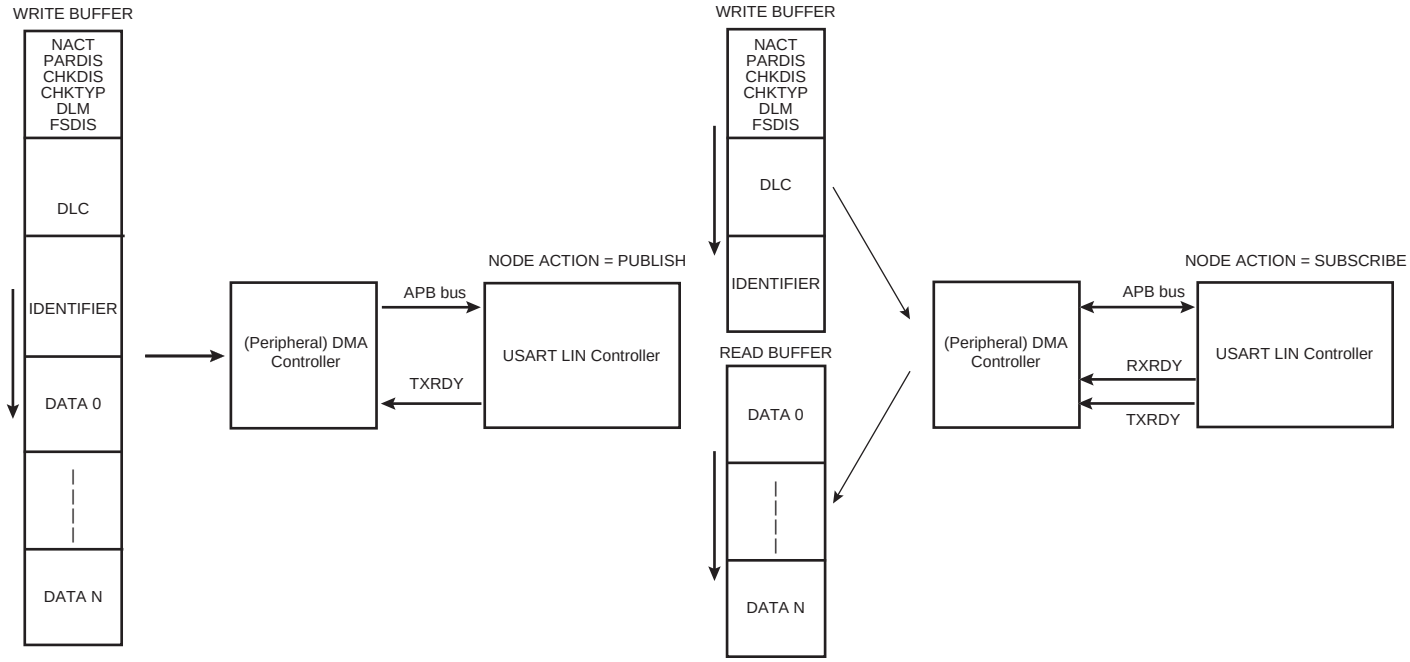
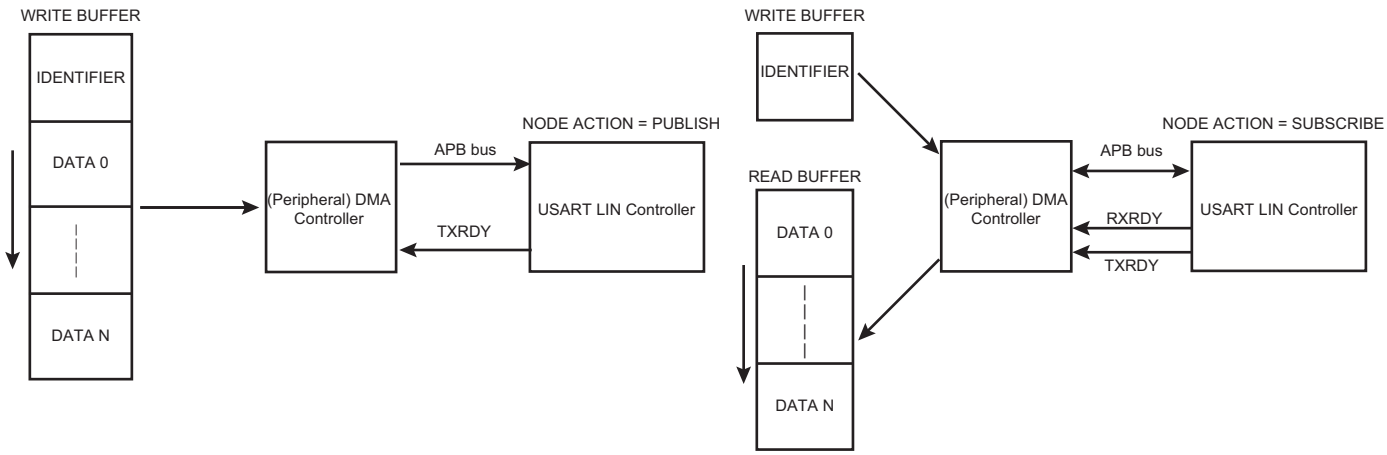


Figure 45-52. Master Node with DMAC (PDCM = 0)



Slave Node Configuration

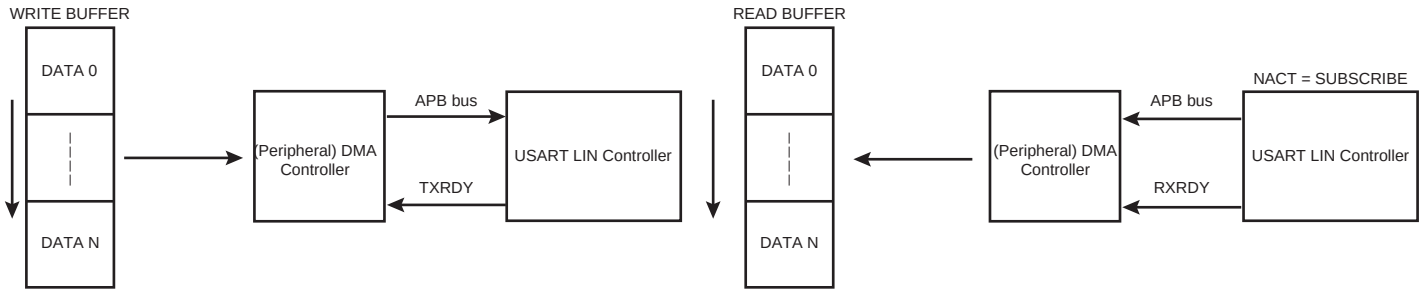
In this configuration, the DMAC transfers only the DATA. The Identifier must be read by the user in the LIN Identifier register (US_LINIR). The LIN mode must be written by the user in US_LINMR.

The WRITE buffer contains the DATA if the USART sends the response (NACT = PUBLISH).

The READ buffer contains the DATA if the USART receives the response (NACT = SUBSCRIBE).

Downloaded from Arrow.com.

Figure 45-53. Slave Node with DMAC



45.6.9.17 Wake-up Request

Any node in a sleeping LIN cluster may request a wake-up.

In the LIN 2.0 specification, the wakeup request is issued by forcing the bus to the dominant state from 250 µs to 5 ms. For this, it is necessary to send the character 0xF0 in order to impose five successive dominant bits. Whatever the baud rate is, this character complies with the specified timings.

- Baud rate min = 1 kbit/s -> t_{bit} = 1 ms -> 5 t_{bit} = 5 ms
- Baud rate max = 20 kbit/s -> t_{bit} = 50 µs -> 5 t_{bit} = 250 µs

In the LIN 1.3 specification, the wakeup request should be generated with the character 0x80 in order to impose eight successive dominant bits.

The user can choose by the WKUPTYP bit in US_LINMR either to send a LIN 2.0 wakeup request (WKUPTYP = 0) or to send a LIN 1.3 wakeup request (WKUPTYP = 1).

A wake-up request is transmitted by writing a 1 to the LINWKUP bit in the US_CR. Once the transfer is completed, the LINTC flag is asserted in the Status register (US_SR). It is cleared by writing a 1 to the RSTSTA bit in the US_CR.

45.6.9.18 Bus Idle Time-out

If the LIN bus is inactive for a certain duration, the slave nodes shall automatically enter in Sleep mode. In the LIN 2.0 specification, this time-out is fixed at 4 seconds. In the LIN 1.3 specification, it is fixed at 25,000 t_{bit}.

In slave Node configuration, the receiver time-out detects an idle condition on the RXD line. When a time-out is detected, the bit TIMEOUT in US_CSR rises and can generate an interrupt, thus indicating to the driver to go into Sleep mode.

The time-out delay period (during which the receiver waits for a new character) is programmed in the TO field of US_RTOR. If a 0 is written to the TO field, the Receiver Time-out is disabled and no time-out is detected. The TIMEOUT bit in US_CSR remains at 0. Otherwise, the receiver loads a 17-bit counter with the value programmed in TO. This counter is decremented at each bit period and reloaded each time a new character is received. If the counter reaches 0, the TIMEOUT bit in the US_CSR rises.

If STTTO is performed, the counter clock is stopped until a first character is received.

Downloaded from Arrow.com.

If RETTO is performed, the counter starts counting down immediately from the value TO.

Table 45-16. Receiver Time-out Programming

LIN Specification	Baud Rate	Time-out period	US_RTOR.TO
2.0	1,000 bit/s	4 s	4,000
	2,400 bit/s		9,600
	9,600 bit/s		38,400
	19,200 bit/s		76,800
	20,000 bit/s		80,000
1.3	–	25,000 t_{bit}	25,000

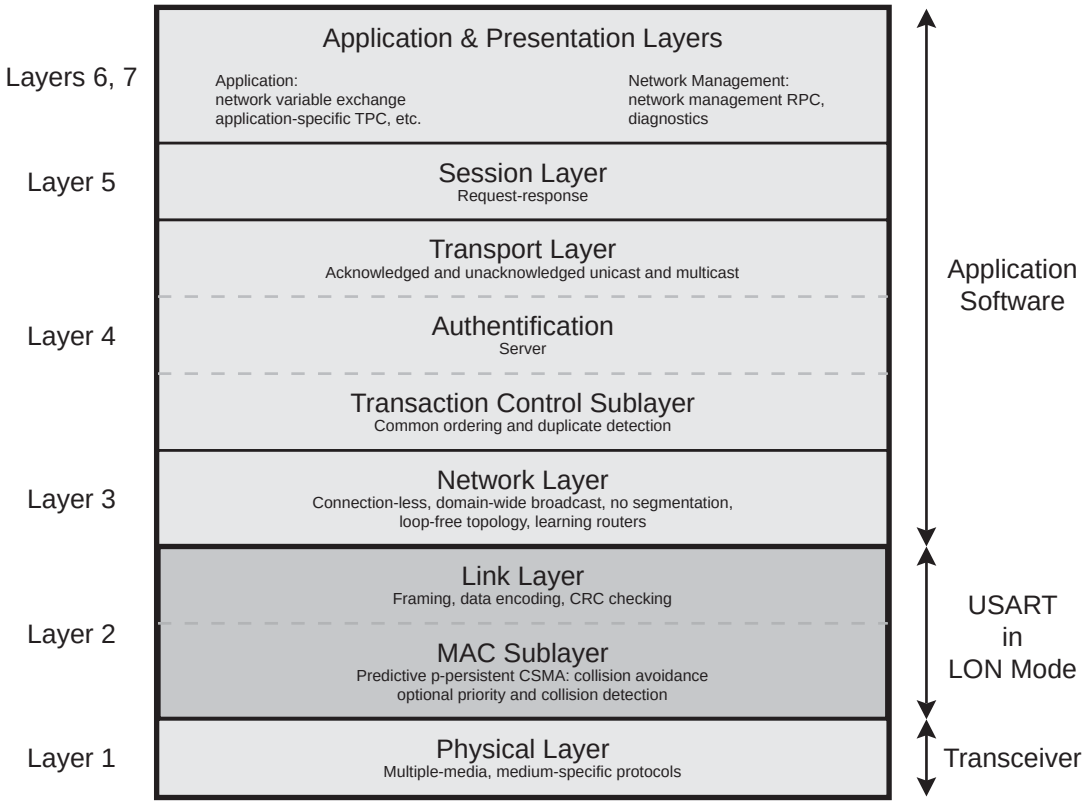
45.6.10 LON Mode

The LON mode provides connectivity to the local operating network (LON).

The LON standard covers all seven layers of the OSI (Open Systems Interconnect) reference model from the physical interfaces such as wired, power line, RF, and IP to the application layer and all layers in between. It was designed from the bottom up as a controls communication platform.

The LON mode enables the transmission and reception of Physical Protocol Data Unit (PPDU) frames with minimum intervention from the microprocessor.

Figure 45-54. LON Protocol Layering



The USART configured in LON mode is a full-layer 2 implementation including standard timings handling, framing (transmit and receive PPDU frames), backlog estimation and other features. At the frame encoding/decoding level, differential Manchester encoding is used (also known as CDP).

Downloaded from Arrow.com.

45.6.10.1 Mode of Operation

To configure the USART to act as a LON node, the USART_MODE field of the US_MR must be set to 0x9.

To avoid unpredictable behavior, any change of the LON node configuration must be preceded by a software reset of the transmitter and the receiver (except the initial node configuration after a hardware reset) and followed by a transmitter/receiver enable. See Section 45.6.10.2.

45.6.10.2 Receiver and Transmitter Control

See Section 45.6.2 "Receiver and Transmitter Control".

45.6.10.3 Character Transmission

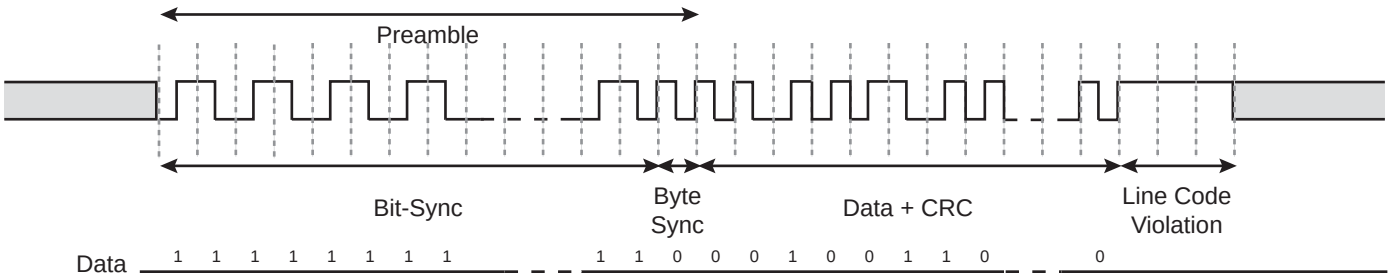
A LON frame is made up of a preamble, a data field (up to 256 bytes) and a 16-bit CRC field. The preamble and CRC fields are automatically generated and the LON node starts the transmission algorithm upon US_LONL2HDR register write. See Section "Sending A Frame".

45.6.10.4 Character Reception

When receiving a LON frame, the Receive Holding register (US_RHR) is updated upon completed character reception and the RXRDY bit in the Status register rises. If a character is completed while the RXRDY bit is set, the OVRE (Overrun Error) bit is set. The LON preamble field is only used for synchronization, therefore only the Data and CRC fields are transmitted to the Receive Holding register (US_RHR). See "Receiving A Frame" .

45.6.10.5 LON Frame

Figure 45-55. LON Framing



Encoding / Decoding

The USART configured in LON mode encodes transmitted data and decodes received data using differential Manchester encoding. In differential Manchester encoding, a '1' bit is indicated by making the first half of the signal equal the last half of the previous bit's signal (no transition at the start of the bit-time). A '0' bit is indicated by making the first half of the signal opposite to the last half of the previous bit's signal (a zero bit is indicated by a transition at the beginning of the bit-time). As is the case with normal Manchester encoding, missing transition at the middle of bit-time represents a Manchester code violation.
The RXIDLEV bit in US_MAN informs the USART of the receiver line idle state value (receiver line inactive) thus ensuring higher reliability of preamble synchronization. By default, RXIDLEV is set to 1 (receiver line is at level 1 when there is no activity).

Differential Manchester encoding is polarity insensitive.

Figure 45-56. LON PPDU



Downloaded from Arrow.com.

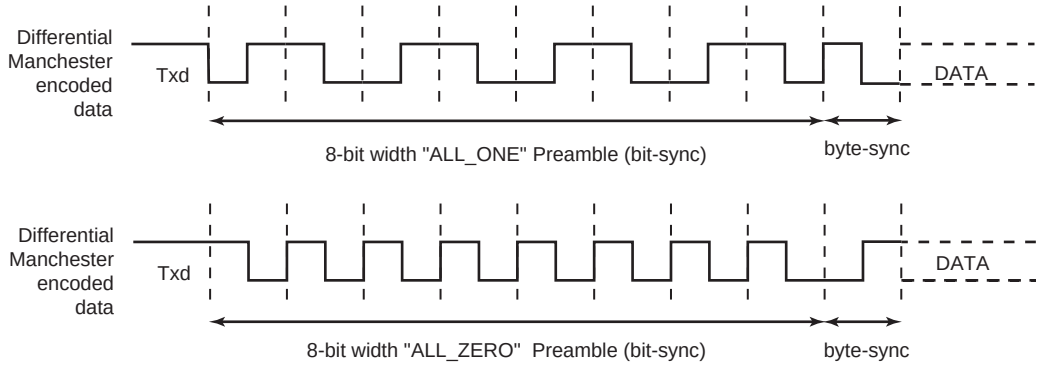
Preamble Transmission

Each LON frame begins with a preamble of variable length which consists of a bit-sync field and a byte-sync field. The LONPL field of the USART LON Preamble register (US_LONPR) defines the preamble length, please note that preamble length of ‘0’ is not allowed.

The LON implementation allows two different preamble patterns ALL_ONE and ALL_ZERO which can be configured through the TX_PL field of the USART Manchester Configuration register (US_MAN). Figure 45-57, ”Preamble Patterns” illustrates and defines the valid patterns.

Other preamble patterns are not supported.

Figure 45-57. Preamble Patterns



Preamble Reception

LON received frames begin with a preamble of variable length. The receiving algorithm does not check the preamble length, although a minimum of length of 4 bits is required for the receiving algorithm to consider the received preamble as valid.

As is the case with LON preamble transmission, two preamble patterns (ALL_ONE and ALL_ZERO) are allowed and can be configured through the RX_PL field of the USART Manchester Configuration register (US_MAN). Figure 45-57 illustrates and defines the valid patterns.

Other preamble patterns are not supported.

Header Transmission

Each LON frame, after sending the preamble, starts with the frame header also called L2HDR according to the CEA-709 specification. This header consist of the priority bit, the alternative path bit and the backlog increment. It is the first data to be sent.

In LON mode the transmitting algorithm starts when the US_LONL2HDR register is written (it is the first data to send).

Header Reception

Each LON frame, after receiving the preamble, receives the frame header also called L2HDR according to the CEA-709 specification. This header consists of the priority bit, the alternative path bit, and the backlog increment.

The frame header is the first received data and the RXRDY bit rises as soon as the frame header as been received and stored in the Receive Holding register (US_RHR).

Data

Data are sent/received serially after the preamble transmission/reception. Data can be either sent/received MSB first or LSB first depending on the MSBF bit value in the US_MR.

CRC

The two last bytes of LON frames are dedicated to CRC.

Downloaded from Arrow.com.

When transmitting, the CRC of the frame is automatically generated and sent when expected.

When receiving frames the CRC is automatically checked and a LCRCE flag is set in US_CSR if the calculated CRC do not match the received one. Note that the two received CRC bytes are seen as two additional data from the user point of view.

End Of Frame

The USART configured in LON mode terminates the frame with a 3 t_{bit} long Manchester code violation. After sending the last CRC bit it maintains the data transitionless during three bit periods.

45.6.10.6 LON Operating Modes

Transmitting/Receiving Modules

According to the LON node configuration and LON network state, the transmitting module will be activated if a transmission request has been made and access to the LON bus granted. It returns to idle state once the transmission ends.

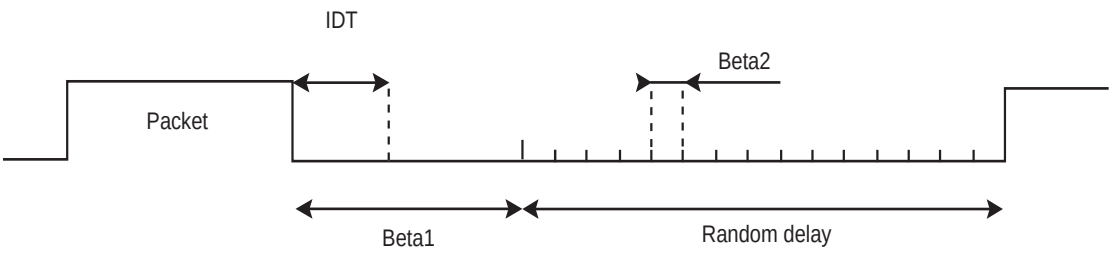
According to the LON node configuration and LON network state, the receiving module will be activated if a valid preamble is detected and the transmitting module is not activated.

comm_type

In the CEA-709 standard, two communication configurations are defined and configurable through the *comm_type* variable. The *comm_type* variable value can be set in the USART LON Mode Register (US_LONMR) through the COMMT bit. The selection of the *comm_type* determines the MAC behavior in the following ways:

- comm_type=1:
  - An indeterminate time is defined during the Beta 1 period in which all transitions on the channel are ignored (Figure 45-58).
  - The MAC sublayer ignores collisions occurring during the first 25% of the transmitted preamble. It optionally (according to the CDTAIL bit of US_LONMR) ignores collisions reported following the transmission of the CRC but prior to the end of transmission.
  - If a collision is detected during preamble transmission, the MAC sublayer can terminate the packet if so configured according to the TCOL bit of the US_LONMR. Collisions detected after the preamble has been sent do not terminate transmission.
- comm_type=2:
  - No indeterminate time is defined at the MAC sublayer.
  - The MAC sublayer shall always terminate the packet upon notification of a collision.

Figure 45-58. LON Indeterminate Time



Collision Detection

As an option of the CEA-709 standard, collision detection is supported through an active low Collision Detect (CD) input from the transceiver.

The Collision Detection source can be either external (See Section 45.4 "I/O Lines Description") or internal. The collision detection source selection is defined through the LCDS bit in the USART LON Mode Register.

Downloaded from Arrow.com.

The Collision Detection feature can be activated through the COLDET bit of the USART LON Mode register (US_LONMR). If the collision detection feature is enabled and CD signal goes low for at least half t_{bit} period then a collision is detected and reported as defined in “comm_type” on page 1180.

Collision Detection Mode.

As defined in “comm_type” on page 1180, if *comm_type=1* the LON node can be either configured to not terminate transmission upon collision notification during preamble transmission or terminate transmission.

The TCOL bit of the US_LONMR allows to decide whether to terminate transmission or not upon collision notification during preamble transmission.

Collision Detection After CRC

As defined in “comm_type” on page 1180, if *comm_type=1* the LON node can be either be configured to ignore collision after the CRC has been sent but prior to the end of the frame.

The CDTAIL bit of the US_LONMR allows to decide whether such collision notifications must be considered or not.

Random Number Generation

The Predictive *p*-persistent CSMA algorithm defined in the CEA-709.1 Standard is based on a random number generation.

This random number is automatically generated by an internal algorithm.

In addition, a USART IC DIFF register (US_ICDIFF) is available to avoid that two same chips with the same software generate the same random number after reset. The value of this register is used by the internal algorithm to generate the random number. Therefore, putting a different value here for each chip ensures that the random number generated after a reset at the same time, will not be the same. It is recommended to put the chip ID code here.

45.6.10.7 LON Node Backlog Estimation

As defined in the CEA-709 standard, the LON node maintains its own backlog estimation. The node backlog estimation is initially set to 1, will always be greater than 1 and will never exceed 63. If the node backlog estimation exceeds the maximum backlog value, the backlog value is set to 63 and a backlog overflow error flag is set (LBLOVFE flag).

The node backlog estimation is incremented each time a frame is sent or received successfully. The increment to the backlog is encoded into the link layer header, and represents the number of messages that the packet shall cause to be generated upon reception.

The backlog decrements under one of the following conditions:

- On waiting to transmit: If Wbase randomizing slots go by without channel activity.
- On receive: If a packet is received with a backlog increment of ‘0’.
- On transmit: If a packet is transmitted with a backlog increment of ‘0’.
- On idle: If a packet cycle time expires without channel activity.

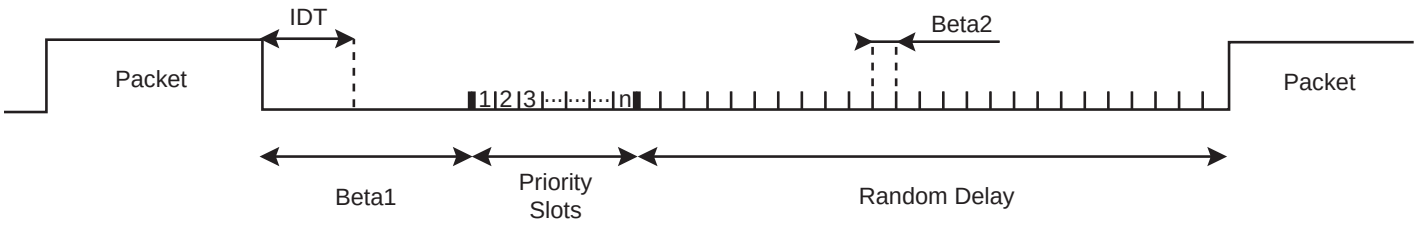
Optional Collision Detection Feature And Backlog Estimation

Each time a frame is transmitted and a collision occurred, the backlog is incremented by 1. In this case, the backlog increment encoded in the link layer is ignored.

Downloaded from Arrow.com.

45.6.10.8 LON Timings

Figure 45-59. LON timings



Beta2

A node wishing to transmit generates a random delay T. This delay is an integer number of randomizing slots of duration Beta2.

The beta2 length (in t_{bit}) is configurable through US_FIDI. Note that a length of '0' is not allowed.

Beta1 Tx/Rx

Beta1 is the period immediately following the end of a packet cycle (see Figure 45-59). A node attempting to transmit monitors the state of the channel, and if it detects no transmission during the Beta1 period, it determines the channel to be idle.

The Beta1 value is different depending on the previous packet type (received packet or transmitted packet).

Beta1Rx and Beta1Tx length can be configured respectively through the USART LON Beta1 Rx register (US_LONB1RX) and the USART LON Beta1 Tx register (US_LONB1TX). Note that a length of '0' is not allowed.

Pcycle Timer

The packet cycle timer is reset to its initial value whenever the backlog is changed. It is started (begins counting down at its current value) whenever the MAC layer becomes idle. An idle MAC layer is defined as:

- Not receiving
- Not transmitting,
- Not waiting to transmit,
- Not timing Beta1,
- Not waiting for priority slots, and not waiting for the first Wbase randomizing window to complete.

On transition from idle to either transmit or receive, the packet cycle timer is halted.

The pcycle timer value can be configured in US_TTGR. Note that '0' value is not allowed.

Wbase

The wbase timer represents the base windows size. Its duration, derived from Beta2, equals 16 Beta2 slots.

Priority Slots

On a channel by channel basis, the protocol supports optional priority. Priority slots, if any, follow immediately after the Beta1 period that follows the transmission of a packet (see Figure 45-59). The number of priority slots per channel ranges from 0 to 127.

The number of priority slots in the LON network configuration is defined through the PSNB field of the USART LON Priority register (US_LONPRIO). And the priority slot affected to the LON node, if any, is defined through the NPS field of the US_LONPRIO register.

Indeterminate Time

See "comm_type" on page 1180.

Like Beta1, the IDT value is different depending on what was the previous frame (transmitted or received frame).

Downloaded from Arrow.com.

IDTRx and IDTTx can be configured respectively through the USART LON IDT Rx register (US_LONIDTRX) and the USART LON IDT Tx register (US_LONIDTTX).

End of Frame Condition

The USART configured in LON mode terminates the frame with a 3 t_{bit} long Manchester code violation. After sending the last CRC bit, it maintains the data transitionless during three bit periods.

While receiving data the USART configured in LON mode will detect an end of frame condition after a t_{eof} transitionless Manchester code violation. The EOFS field in the USART LON Mode Register can configure t_{eof}.

45.6.10.9 LON Errors

All these flags can be read in the LON Channel Status register (US_CSR) and will generate interrupts if configured in the LON Interrupt Enable register (US_IER).

These flag can be reset through the RSTSTA bit in US_CR.

Underrun Error

If the USART is in LON mode and if a character is sent while the Transmit Holding register (US_THR) is empty, the UNRE bit flag is set.

Collision Detection

The LCOL flag is set whenever a valid collision has been detected and the LON node is configured to report it (see “Collision Detection” on page 1180).

LON Frame Early Termination

The LFET flag is set whenever a LON frame has been terminated early due to collision detection.

Reception Error

The LCRCE flag is set if the received frame has an erroneous CRC and the flag LSFE is set if the received frame is too short (LON frames must be at least 8 bytes long).

These flags can be read in US_CSR.

Backlog Overflow

The LBLOVFE flag is set if the LON node backlog estimation goes over 63 which is the maximum backlog value.

45.6.10.10 Drift Compensation

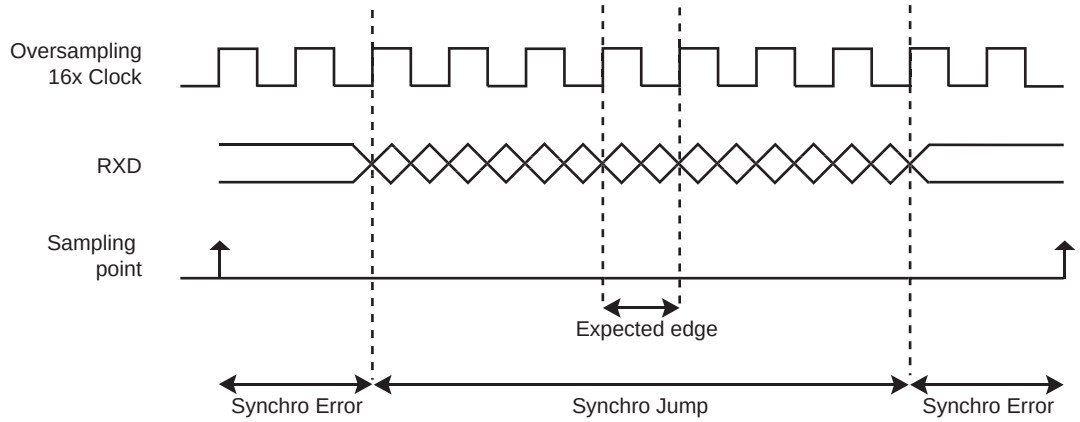
It may happen that while receiving a frame the baud rate used by the sender is not exactly the one expected due to sender clock drifting for instance, in such case the hardware drift compensation algorithm allows to recover up to 16% clock drift (expected baud rate ±16% will be supported).

Drift compensation is available only in 16X Oversampling mode. To enable the hardware system, the DRIFT bit of the USART_MAN register must be set. If the RXD edge is between 1 and 3 16X clock cycle far from the expected edge, then the period is shortened or lengthened accordingly to center the RXD edge.

Drift compensation hardware feature allows up to 16% clock drift to be handled, provided system clock is fast enough compared to the selected baud rate.

Downloaded from Arrow.com.

Figure 45-60. Bit Resynchronization

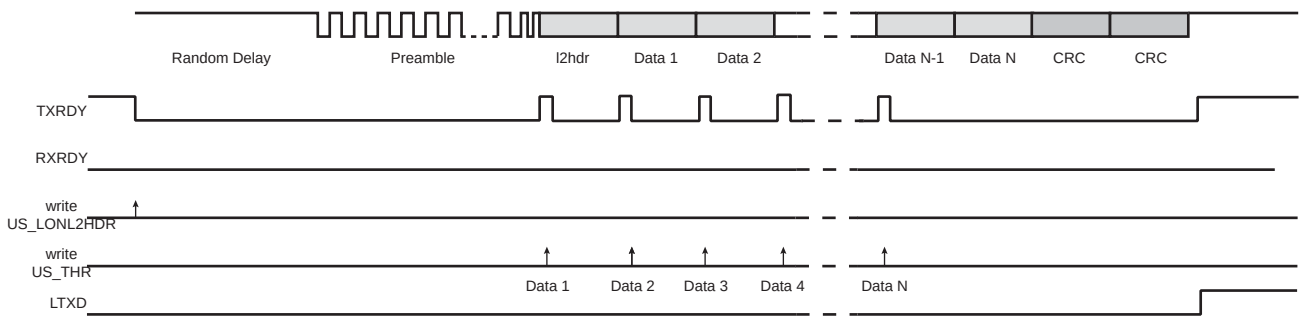


45.6.10.11 LON Frame Handling

Sending A Frame

1. Write TXEN and RXEN in US_CR to enable both the transmitter and the receiver.
2. Write USART_MODE in US_MR to select the LON mode configuration.
3. Write CD and FP in US_BRGR to configure the baud rate.
4. Write COMMT, COLDET, TCOL, CDTAIL, RDMNBM and DMAM in US_LONMR to configure the LON operating mode.
5. Write BETA2, BETA1TX, BETA1RX, PCYCLE, PSNB, NPS, IDTTX and ITDRX respectively in US_FIDI, US_LONB1TX, US_LONB1RX, US_TTGR, US_LONPRIO, US_LONIDTTX and US_LONIDTRX to set the LON network configuration.
6. Write TX_PL in US_MAN to select the preamble pattern to use.
7. Write LONPL and LONDL in US_LONPR and US_LONDL to set the frame transfer.
8. Check that TXRDY in US_CSR is set to 1.
9. Write US_LONL2HDR register to send the header.
10. Wait until TXRDY in US_CSR rises.
11. Write TCHR in US_THR to send a byte.
12. If all the data have not been written, redo the two previous steps.
13. Wait until LTXD in US_CSR rises.
14. Check the LON errors.

Figure 45-61. Tx Frame

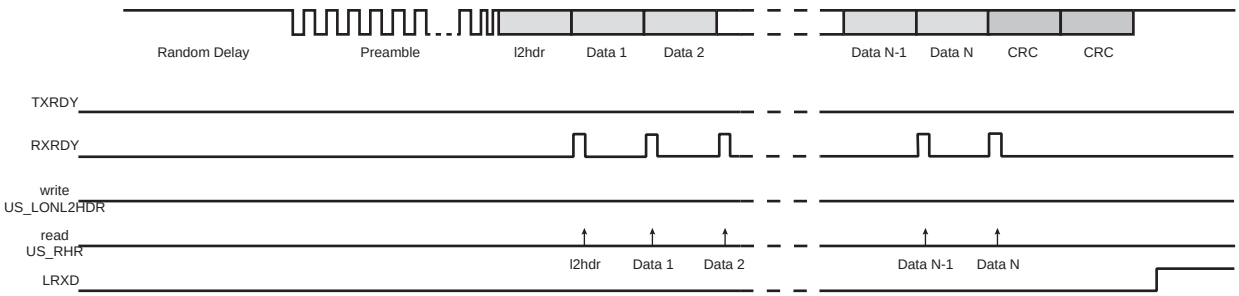


Downloaded from Arrow.com.

Receiving A Frame

1. Write TXEN and RXEN in US_CR to enable both the transmitter and the receiver.
2. Write USART_MODE in US_MR to select the LON mode configuration.
3. Write CD and FP in US_BRGR to configure the baud rate.
4. Write COMMT, COLDET, TCOL, CDTAIL, RDMNBM and DMAM in US_LONMR to configure the LON operating mode.
5. Write BETA2, BETA1TX, BETA1RX, PCYCLE, PSNB, NPS, IDTTX and ITDRX respectively in US_FIDI, US_LONB1TX, US_LONB1RX, US_TTGR, US_LONPRIO, US_LONIDTTX and US_LONIDTRX to set the LON network configuration.
6. Write RXIDLEV and RX_PL in US_MAN to indicate the receiver line value and select the preamble pattern to use.
7. Wait until RXRDY in US_CSR rises.
8. Read RCHR in US_RHR.
9. If all the data and the two CRC bytes have not been read, redo the two previous steps.
10. Wait until LRXD in US_CSR rises.
11. Check the LON errors.

Figure 45-62. Rx Frame



45.6.10.12 LON Frame Handling with the Peripheral DMA Controller

The USART can be used in association with the DMA Controller in order to transfer data directly into/from the on- and off-chip memories without any processor intervention.
The DMA uses the trigger flags, TXRDY and RXRDY, to write or read into the USART. The DMA always writes in the Transmit Holding register (US_THR) and it always reads in the Receive Holding register (US_RHR). The size of the data written or read by the DMA in the USART is always a byte.

Configuration

The user can choose between two DMA modes by the DMAM bit in the LON Mode register (US_LONMR):

- DMAM = 1: The LON frame data length (DATAL) is stored in the WRITE buffer and it is written by the DMA in the Transmit Holding register US_THR (instead of the LON Data Length register US_LONDL).
- DMAM = 0: The LON frame data length (DATAL) is not stored in the WRITE buffer and it must be written by the user in the LON Data Length register (US_LONDL).

In both DMA modes L2HDR is considered as a data and its value must be stored in the WRITE buffer as the first data to write.

Downloaded from Arrow.com.

Figure 45-63. DMAM = 1

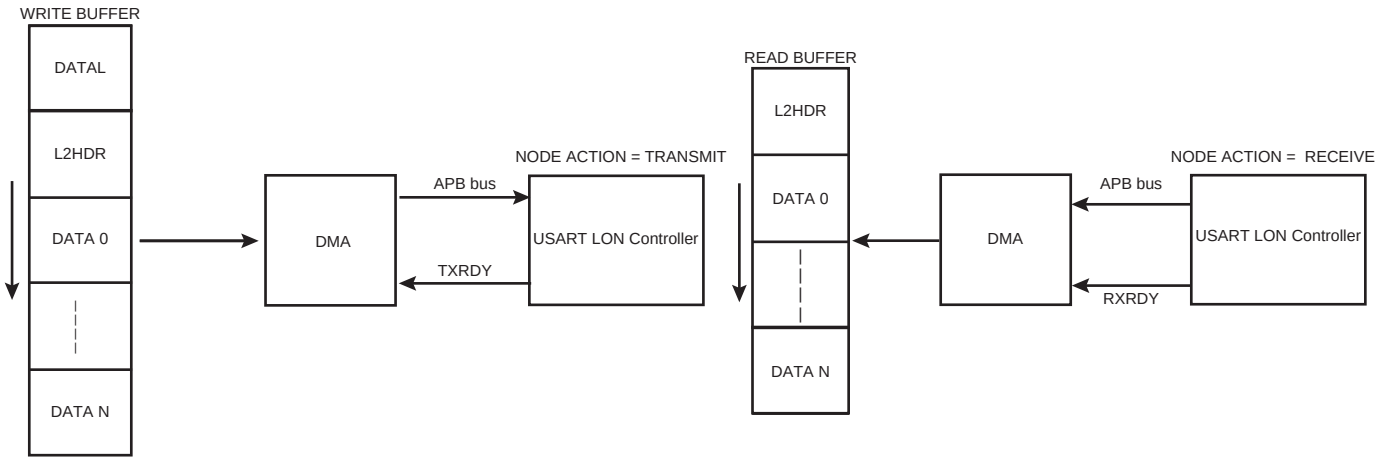
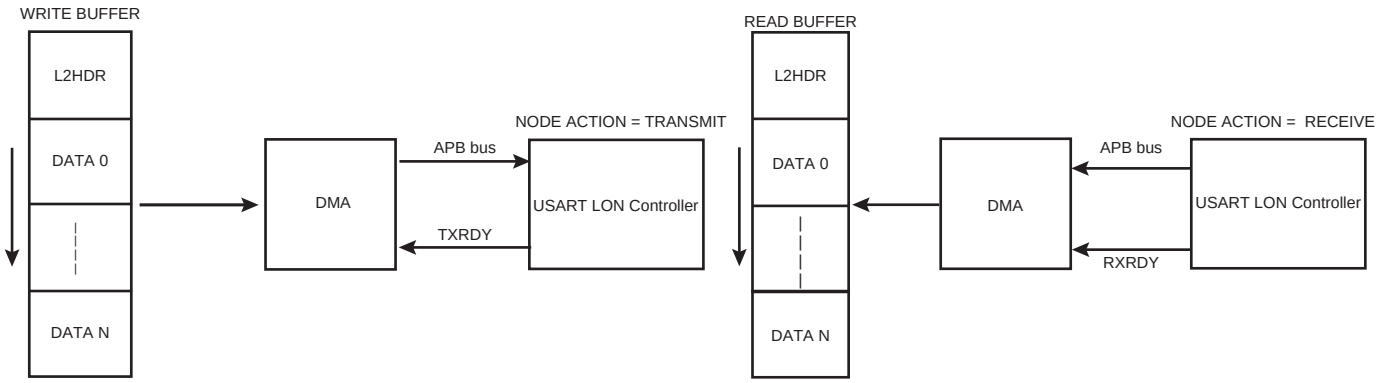


Figure 45-64. DMAM = 0



DMA and Collision Detection

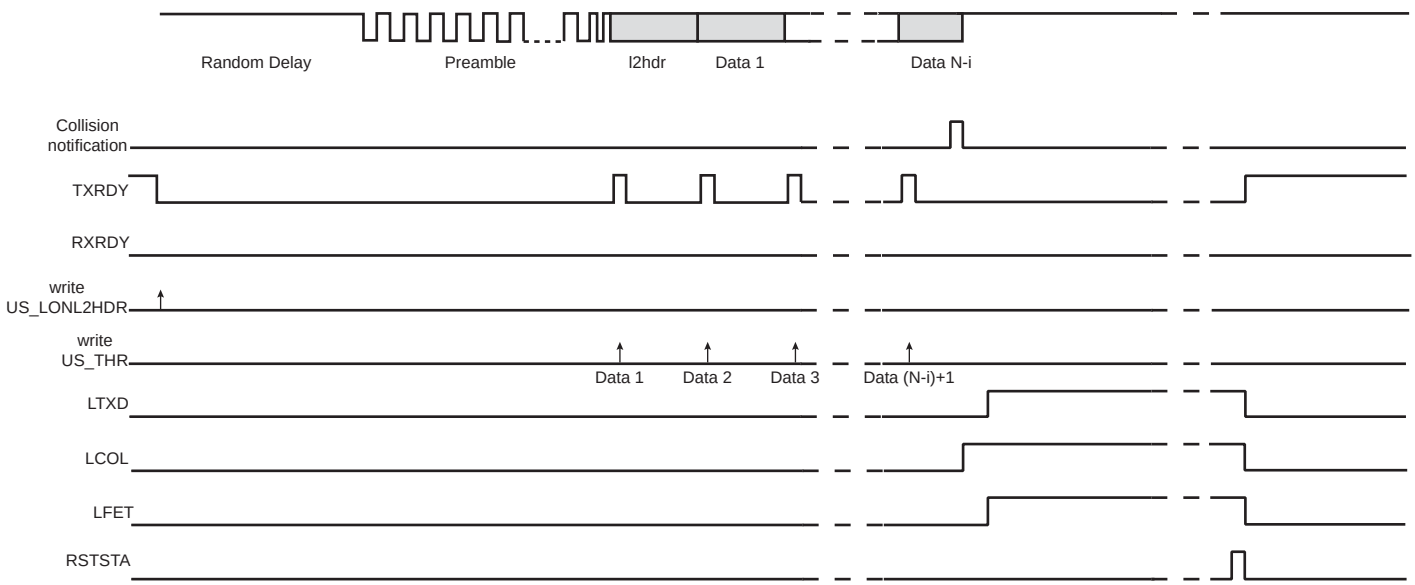
As explained in “comm_type” on page 1180, depending on LON configuration the transmission may be terminated early upon collision notification which means that the DMA transfer may be stopped before its end.

In case of early end of transmission due to collision detection the USART in LON mode acts as follows:

- Send the end of frame trigger.
- Hold down TXRDY avoiding thus any additional DMA transfer.
- Set LTXD, LCOL and LFET flags in US_CSR.
- Wait that the application reconfigure the DMA.
- Wait until LCOL and LFET flags are cleared through the RSTSTA bit of the US_CR (it will release the TXRDY signal).

Downloaded from Arrow.com.

Figure 45-65. DMA, Collision and Early Frame Termination



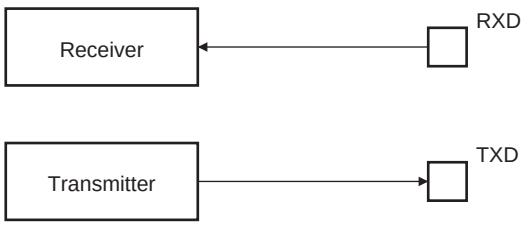
45.6.11 Test Modes

The USART can be programmed to operate in three different test modes. The internal loopback capability allows on-board diagnostics. In Loopback mode, the USART interface pins are disconnected or not and reconfigured for loopback internally or externally.

45.6.11.1 Normal Mode

Normal mode connects the RXD pin on the receiver input and the transmitter output on the TXD pin.

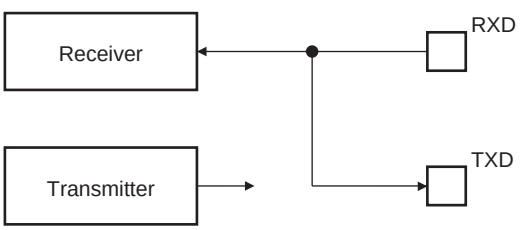
Figure 45-66. Normal Mode Configuration



45.6.11.2 Automatic Echo Mode

Automatic Echo mode allows bit-by-bit retransmission. When a bit is received on the RXD pin, it is sent to the TXD pin, as shown in Figure 45-67. Programming the transmitter has no effect on the TXD pin. The RXD pin is still connected to the receiver input, thus the receiver remains active.

Figure 45-67. Automatic Echo Mode Configuration

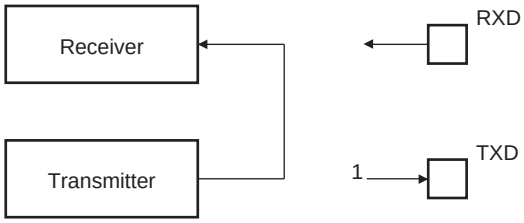


Downloaded from Arrow.com.

45.6.11.3 Local Loopback Mode

Local Loopback mode connects the output of the transmitter directly to the input of the receiver, as shown in Figure 45-68. The TXD and RXD pins are not used. The RXD pin has no effect on the receiver and the TXD pin is continuously driven high, as in idle state.

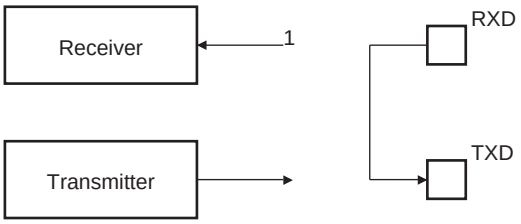
Figure 45-68. Local Loopback Mode Configuration



45.6.11.4 Remote Loopback Mode

Remote Loopback mode directly connects the RXD pin to the TXD pin, as shown in Figure 45-69. The transmitter and the receiver are disabled and have no effect. This mode allows bit-by-bit retransmission.

Figure 45-69. Remote Loopback Mode Configuration



Downloaded from Arrow.com.

45.6.12 Register Write Protection

To prevent any single software error from corrupting USART behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the USART Write Protection Mode Register (US_WPMR).

If a write access to a write-protected register is detected, the WPVS flag in the USART Write Protection Status Register (US_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading the US_WPSR.

The following registers can be write-protected:

- USART Mode Register
- USART Baud Rate Generator Register
- USART Receiver Time-out Register
- USART Transmitter Timeguard Register
- USART FI DI RATIO Register
- USART IrDA Filter Register
- USART Manchester Configuration Register
- USART LON Mode Register
- USART LON Beta1 Tx Register
- USART LON Beta1 Rx Register
- USART LON Priority Register
- USART LON IDT Tx Register
- USART LON IDT Rx Register
- USART IC DIFF Register

Downloaded from Arrow.com.

45.7 Universal Synchronous Asynchronous Receiver Transmitter (USART) User Interface

Table 45-17. Register Mapping

Offset	Register	Name	Access	Reset
0x0000	Control Register	US_CR	Write-only	–
0x0004	Mode Register	US_MR	Read/Write	0x0
0x0008	Interrupt Enable Register	US_IER	Write-only	–
0x000C	Interrupt Disable Register	US_IDR	Write-only	–
0x0010	Interrupt Mask Register	US_IMR	Read-only	0x0
0x0014	Channel Status Register	US_CSR	Read-only	0x0
0x0018	Receive Holding Register	US_RHR	Read-only	0x0
0x001C	Transmit Holding Register	US_THR	Write-only	–
0x0020	Baud Rate Generator Register	US_BRGR	Read/Write	0x0
0x0024	Receiver Time-out Register	US_RTOR	Read/Write	0x0
0x0028	Transmitter Timeguard Register	US_TTGR	Read/Write	0x0
0x002C–0x003C	Reserved	–	–	–
0x0040	FI DI Ratio Register	US_FIDI	Read/Write	0x174
0x0044	Number of Errors Register	US_NER	Read-only	0x0
0x0048	Reserved	–	–	–
0x004C	IrDA Filter Register	US_IF	Read/Write	0x0
0x0050	Manchester Configuration Register	US_MAN	Read/Write	0xB0011004
0x0054	LIN Mode Register	US_LINMR	Read/Write	0x0
0x0058	LIN Identifier Register	US_LINIR	Read/Write(1)	0x0
0x005C	LIN Baud Rate Register	US_LINBRR	Read-only	0x0
0x0060	LON Mode Register	US_LONMR	Read/Write	0x0
0x0064	LON Preamble Register	US_LONPR	Read/Write	0x0
0x0068	LON Data Length Register	US_LONDL	Read/Write	0x0
0x006C	LON L2HDR Register	US_LONL2HDR	Read/Write	0x0
0x0070	LON Backlog Register	US_LONBL	Read	0x0
0x0074	LON Beta1 Tx Register	US_LONB1TX	Read/Write	0x0
0x0078	LON Beta1 Rx Register	US_LONB1RX	Read/Write	0x0
0x007C	LON Priority Register	US_LONPRIO	Read/Write	0x0
0x0080	LON IDT Tx Register	US_IDTTX	Read/Write	0x0
0x0084	LON IDT Rx Register	US_IDTRX	Read/Write	0x0
0x0088	IC DIFF Register	US_ICDIFF	Read/Write	0x0
0x008C–0x00E0	Reserved	–	–	–
0x00E4	Write Protection Mode Register	US_WPMR	Read/Write	0x0
0x00E8	Write Protection Status Register	US_WPSR	Read-only	0x0
0x00EC–0x00FC	Reserved	–	–	–

Notes: 1. Write is possible only in LIN master node configuration.

Downloaded from Arrow.com.

45.7.1 USART Control Register

Name: US_CR

Address: 0x40024000 (0), 0x40028000 (1), 0x4002C000 (2)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	LINWKUP	LINABT	RTSDIS	RTSEN	DTRDIS	DTREN

15	14	13	12	11	10	9	8
RETTO	RSTNACK	RSTIT	SENDA	STTTO	STPBRK	STTBRK	RSTSTA

7	6	5	4	3	2	1	0
TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	–	–

For SPI control, see Section 45.7.2 ”USART Control Register (SPI_MODE)”.

- **RSTRX: Reset Receiver**

0: No effect.

1: Resets the receiver.

- **RSTTX: Reset Transmitter**

0: No effect.

1: Resets the transmitter.

- **RXEN: Receiver Enable**

0: No effect.

1: Enables the receiver, if RXDIS is 0.

- **RXDIS: Receiver Disable**

0: No effect.

1: Disables the receiver.

- **TXEN: Transmitter Enable**

0: No effect.

1: Enables the transmitter if TXDIS is 0.

- **TXDIS: Transmitter Disable**

0: No effect.

1: Disables the transmitter.

- **RSTSTA: Reset Status Bits**

0: No effect.

1: Resets the status bits PARE, FRAME, OVRE, MANERR, LINBE, LINISFE, LINIPE, LINCE, LINSNRE, LINSTE, LINHTE, LINID, LINTC, LINBK and RXBRK in US_CSR.

Downloaded from Arrow.com.

- **STTBRK: Start Break**

0: No effect.

1: Starts transmission of a break after the characters present in US_THR and the Transmit Shift Register have been transmitted. No effect if a break is already being transmitted.

- **STPBRK: Stop Break**

0: No effect.

1: Stops transmission of the break after a minimum of one character length and transmits a high level during 12-bit periods. No effect if no break is being transmitted.

- **STTTO: Clear TIMEOUT Flag and Start Time-out After Next Character Received**

0: No effect.

1: Starts waiting for a character before enabling the time-out counter. Immediately disables a time-out period in progress. Resets the status bit TIMEOUT in US_CSR.

- **SENDA: Send Address**

0: No effect.

1: In Multidrop mode only, the next character written to the US_THR is sent with the address bit set.

- **RSTIT: Reset Iterations**

0: No effect.

1: Resets ITER in US_CSR. No effect if the ISO7816 is not enabled.

- **RSTNACK: Reset Non Acknowledge**

0: No effect

1: Resets NACK in US_CSR.

- **RETTO: Start Time-out Immediately**

0: No effect

1: Immediately restarts time-out period.

- **DTREN: Data Terminal Ready Enable**

0: No effect.

1: Drives the pin DTR to 0.

- **DTRDIS: Data Terminal Ready Disable**

0: No effect.

1: Drives the pin DTR to 1.

- **RTSEN: Request to Send Pin Control**

0: No effect.

1: Drives RTS pin to 1 if US_MR.USART_MODE field = 2, else drives RTS pin to 0 if US_MR.USART_MODE field = 0.

Downloaded from Arrow.com.

- **RTSDIS: Request to Send Pin Control**

0: No effect.

1: Drives RTS pin to 0 if US_MR.USART_MODE field = 2 (if PDC RX buffer is not full), else drives RTS pin to 1 if US_MR.USART_MODE field = 0.

- **LINABT: Abort LIN Transmission**

0: No effect.

1: Abort the current LIN transmission.

- **LINWKUP: Send LIN Wakeup Signal**

0: No effect.

1: Sends a wakeup signal on the LIN bus.

Downloaded from Arrow.com.

45.7.2 USART Control Register (SPI_MODE)

Name: US_CR (SPI_MODE)

Address: 0x40024000 (0), 0x40028000 (1), 0x4002C000 (2)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	RCS	FCS	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	RSTSTA
7	6	5	4	3	2	1	0
TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	–	–

This configuration is relevant only if USART_MODE = 0xE or 0xF in the USART Mode Register.

- **RSTRX: Reset Receiver**

0: No effect.

1: Resets the receiver.

- **RSTTX: Reset Transmitter**

0: No effect.

1: Resets the transmitter.

- **RXEN: Receiver Enable**

0: No effect.

1: Enables the receiver, if RXDIS is 0.

- **RXDIS: Receiver Disable**

0: No effect.

1: Disables the receiver.

- **TXEN: Transmitter Enable**

0: No effect.

1: Enables the transmitter if TXDIS is 0.

- **TXDIS: Transmitter Disable**

0: No effect.

1: Disables the transmitter.

- **RSTSTA: Reset Status Bits**

0: No effect.

1: Resets the status bits OVRE, UNRE in US_CSR.

Downloaded from Arrow.com.

- **FCS: Force SPI Chip Select**

Applicable if USART operates in SPI Master mode (USART_MODE = 0xE):

0: No effect.

1: Forces the Slave Select Line NSS (RTS pin) to 0, even if USART is not transmitting, in order to address SPI slave devices supporting the CSAAT mode (Chip Select Active After Transfer).

- **RCS: Release SPI Chip Select**

Applicable if USART operates in SPI Master mode (USART_MODE = 0xE):

0: No effect.

1: Releases the Slave Select Line NSS (RTS pin).

Downloaded from Arrow.com.

45.7.3 USART Mode Register

Name: US_MR

Address: 0x40024004 (0), 0x40028004 (1), 0x4002C004 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
ONEBIT	MODSYNC	MAN	FILTER	–	MAX_ITERATION		
23	22	21	20	19	18	17	16
INVDATA	VAR_SYNC	DSNACK	INACK	OVER	CLKO	MODE9	MSBF
15	14	13	12	11	10	9	8
CHMODE		NBSTOP		PAR			SYNC
7	6	5	4	3	2	1	0
CHRL		USCLKS		USART_MODE			

This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

For SPI configuration, see Section 45.7.4 ”USART Mode Register (SPI_MODE)”.

- **USART_MODE: USART Mode of Operation**

Value	Name	Description
0x0	NORMAL	Normal mode
0x1	RS485	RS485
0x2	HW_HANDSHAKING	Hardware Handshaking
0x3	MODEM	Modem
0x4	IS07816_T_0	IS07816 Protocol: T = 0
0x6	IS07816_T_1	IS07816 Protocol: T = 1
0x8	IRDA	IrDA
0x9	LON	LON
0xE	SPI_MASTER	SPI Master mode (CLKO must be written to 1 and USCLKS = 0, 1 or 2)
0xF	SPI_SLAVE	SPI Slave mode

- **USCLKS: Clock Selection**

Value	Name	Description
0	MCK	Peripheral clock is selected
1	DIV	Peripheral clock divided (DIV=8) is selected
2	PCK	PMC programmable clock (PCK) is selected. If the SCK pin is driven (CLKO = 1), the CD field must be greater than 1.
3	SCK	Serial clock (SCK) is selected

Downloaded from Arrow.com.

- **CHRL: Character Length**

Value	Name	Description
0	5_BIT	Character length is 5 bits
1	6_BIT	Character length is 6 bits
2	7_BIT	Character length is 7 bits
3	8_BIT	Character length is 8 bits

- **SYNC: Synchronous Mode Select**

0: USART operates in Asynchronous mode.

1: USART operates in Synchronous mode.

- **PAR: Parity Type**

Value	Name	Description
0	EVEN	Even parity
1	ODD	Odd parity
2	SPACE	Parity forced to 0 (Space)
3	MARK	Parity forced to 1 (Mark)
4	NO	No parity
6	MULTIDROP	Multidrop mode

- **NBSTOP: Number of Stop Bits**

Value	Name	Description
0	1_BIT	1 stop bit
1	1_5_BIT	1.5 stop bit (SYNC = 0) or reserved (SYNC = 1)
2	2_BIT	2 stop bits

- **CHMODE: Channel Mode**

Value	Name	Description
0	NORMAL	Normal mode
1	AUTOMATIC	Automatic Echo. Receiver input is connected to the TXD pin.
2	LOCAL_LOOPBACK	Local Loopback. Transmitter output is connected to the Receiver Input.
3	REMOTE_LOOPBACK	Remote Loopback. RXD pin is internally connected to the TXD pin.

- **MSBF: Bit Order**

0: Least significant bit is sent/received first.

1: Most significant bit is sent/received first.

- **MODE9: 9-bit Character Length**

0: CHRL defines character length

1: 9-bit character length

Downloaded from Arrow.com.

- **CLKO: Clock Output Select**

0: The USART does not drive the SCK pin.

1: The USART drives the SCK pin if USCLKS does not select the external clock SCK.

- **OVER: Oversampling Mode**

0: 16 × Oversampling

1: 8 × Oversampling

- **INACK: Inhibit Non Acknowledge**

0: The NACK is generated.

1: The NACK is not generated.

- **DSNACK: Disable Successive NACK**

0: NACK is sent on the ISO line as soon as a parity error occurs in the received character (unless INACK is set).

1: Successive parity errors are counted up to the value specified in the MAX_ITERATION field. These parity errors generate a NACK on the ISO line. As soon as this value is reached, no additional NACK is sent on the ISO line. The flag ITER is asserted.

Note: MAX_ITERATION field must be set to 0 if DSNACK is cleared.

- **INVDATA: Inverted Data**

0: The data field transmitted on TXD line is the same as the one written in US_THR or the content read in US_RHR is the same as RXD line. Normal mode of operation.

1: The data field transmitted on TXD line is inverted (voltage polarity only) compared to the value written on US_THR or the content read in US_RHR is inverted compared to what is received on RXD line (or ISO7816 IO line). Inverted mode of operation, useful for contactless card application. To be used with configuration bit MSBF.

- **VAR_SYNC: Variable Synchronization of Command/Data Sync Start Frame Delimiter**

0: User defined configuration of command or data sync field depending on MODSYNC value.

1: The sync field is updated when a character is written into US_THR.

- **MAX_ITERATION: Maximum Number of Automatic Iteration**

0–7: Defines the maximum number of iterations in mode ISO7816, protocol T = 0.

- **FILTER: Receive Line Filter**

0: The USART does not filter the receive line.

1: The USART filters the receive line using a three-sample filter (1/16-bit clock) (2 over 3 majority).

- **MAN: Manchester Encoder/Decoder Enable**

0: Manchester encoder/decoder are disabled.

1: Manchester encoder/decoder are enabled.

- **MODSYNC: Manchester Synchronization Mode**

0:The Manchester start bit is a 0 to 1 transition

1: The Manchester start bit is a 1 to 0 transition.

Downloaded from Arrow.com.

- **ONEBIT: Start Frame Delimiter Selector**

0: Start frame delimiter is COMMAND or DATA SYNC.

1: Start frame delimiter is one bit.

Downloaded from Arrow.com.

45.7.4 USART Mode Register (SPI_MODE)

Name: US_MR (SPI_MODE)

Address: 0x40024004 (0), 0x40028004 (1), 0x4002C004 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	WRDBT	–	CLKO	–	CPOL
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	CPHA
7	6	5	4	3	2	1	0
CHRL		USCLKS		USART_MODE			

This configuration is relevant only if USART_MODE = 0xE or 0xF in the USART Mode Register.
This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

- **USART_MODE: USART Mode of Operation**

Value	Name	Description
0xE	SPI_MASTER	SPI master
0xF	SPI_SLAVE	SPI Slave

- **USCLKS: Clock Selection**

Value	Name	Description
0	MCK	Peripheral clock is selected
1	DIV	Peripheral clock divided (DIV=8) is selected
3	SCK	Serial Clock SLK is selected

- **CHRL: Character Length**

Value	Name	Description
3	8_BIT	Character length is 8 bits

- **CPHA: SPI Clock Phase**
  – Applicable if USART operates in SPI mode (USART_MODE = 0xE or 0xF):

0: Data is changed on the leading edge of SPCK and captured on the following edge of SPCK.

1: Data is captured on the leading edge of SPCK and changed on the following edge of SPCK.

CPHA determines which edge of SPCK causes data to change and which edge causes data to be captured. CPHA is used with CPOL to produce the required clock/data relationship between master and slave devices.

- **CPOL: SPI Clock Polarity**

Applicable if USART operates in SPI mode (Slave or Master, USART_MODE = 0xE or 0xF):

0: The inactive state value of SPCK is logic level zero.

1: The inactive state value of SPCK is logic level one.

Downloaded from Arrow.com.

CPOL is used to determine the inactive state value of the serial clock (SPCK). It is used with CPHA to produce the required clock/data relationship between master and slave devices.

- **CLKO: Clock Output Select**

0: The USART does not drive the SCK pin.

1: The USART drives the SCK pin if USCLKS does not select the external clock SCK.

- **WRDBT: Wait Read Data Before Transfer**

0: The character transmission starts as soon as a character is written into US_THR (assuming TXRDY was set).

1: The character transmission starts when a character is written and only if RXRDY flag is cleared (Receive Holding Register has been read).

Downloaded from Arrow.com.

45.7.5 USART Interrupt Enable Register

Name: US_IER

Address: 0x40024008 (0), 0x40028008 (1), 0x4002C008 (2)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	MANE
23	22	21	20	19	18	17	16
–	–	–	–	CTSIC	DCDIC	DSRIC	RIIC
15	14	13	12	11	10	9	8
–	–	NACK	–	–	ITER	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	RXBRK	TXRDY	RXRDY

For SPI specific configuration, see Section 45.7.6 ”USART Interrupt Enable Register (SPI_MODE)”.

For LIN specific configuration, see Section 45.7.7 ”USART Interrupt Enable Register (LIN_MODE)”.

For LON specific configuration, see Section 45.7.8 ”USART Interrupt Enable Register (LON_MODE)”.

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Enables the corresponding interrupt.

- **RXRDY: RXRDY Interrupt Enable**
- **TXRDY: TXRDY Interrupt Enable**
- **RXBRK: Receiver Break Interrupt Enable**
- **OVRE: Overrun Error Interrupt Enable**
- **FRAME: Framing Error Interrupt Enable**
- **PARE: Parity Error Interrupt Enable**
- **TIMEOUT: Time-out Interrupt Enable**
- **TXEMPTY: TXEMPTY Interrupt Enable**
- **ITER: Max number of Repetitions Reached Interrupt Enable**
- **NACK: Non Acknowledge Interrupt Enable**
- **RIIC: Ring Indicator Input Change Enable**
- **DSRIC: Data Set Ready Input Change Enable**
- **DCDIC: Data Carrier Detect Input Change Interrupt Enable**
- **CTSIC: Clear to Send Input Change Interrupt Enable**

Downloaded from Arrow.com.

- **MANE: Manchester Error Interrupt Enable**

Downloaded from Arrow.com.

45.7.6 USART Interrupt Enable Register (SPI_MODE)

Name: US_IER (SPI_MODE)

Address: 0x40024008 (0), 0x40028008 (1), 0x4002C008 (2)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	NSSE	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	UNRE	TXEMPTY	–
7	6	5	4	3	2	1	0
–	–	OVRE	–	–	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE = 0xE or 0xF in the USART Mode Register.

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Enables the corresponding interrupt.

- **RXRDY: RXRDY Interrupt Enable**
- **TXRDY: TXRDY Interrupt Enable**
- **OVRE: Overrun Error Interrupt Enable**
- **TXEMPTY: TXEMPTY Interrupt Enable**
- **UNRE: SPI Underrun Error Interrupt Enable**
- **NSSE: NSS Line (Driving CTS Pin) Rising or Falling Edge Event Interrupt Enable**

Downloaded from Arrow.com.

45.7.7 USART Interrupt Enable Register (LIN_MODE)

Name: US_IER (LIN_MODE)

Address: 0x40024008 (0), 0x40028008 (1), 0x4002C008 (2)

Access: Write-only

31	30	29	28	27	26	25	24
LINHTE	LINSTE	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
LINTC	LINID	LINBK	–	–	–	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE = 0xA or 0xB in the USART Mode Register.

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Enables the corresponding interrupt.

- **RXRDY: RXRDY Interrupt Enable**
- **TXRDY: TXRDY Interrupt Enable**
- **OVRE: Overrun Error Interrupt Enable**
- **FRAME: Framing Error Interrupt Enable**
- **PARE: Parity Error Interrupt Enable**
- **TIMEOUT: Time-out Interrupt Enable**
- **TXEMPTY: TXEMPTY Interrupt Enable**
- **LINBK: LIN Break Sent or LIN Break Received Interrupt Enable**
- **LINID: LIN Identifier Sent or LIN Identifier Received Interrupt Enable**
- **LINTC: LIN Transfer Completed Interrupt Enable**
- **LINBE: LIN Bus Error Interrupt Enable**
- **LINISFE: LIN Inconsistent Synch Field Error Interrupt Enable**
- **LINIPE: LIN Identifier Parity Interrupt Enable**
- **LINCE: LIN Checksum Error Interrupt Enable**
- **LINSNRE: LIN Slave Not Responding Error Interrupt Enable**

Downloaded from Arrow.com.

- **LINSTE: LIN Synch Tolerance Error Interrupt Enable**
- **LINHTE: LIN Header Timeout Error Interrupt Enable**

Downloaded from Arrow.com.

45.7.8 USART Interrupt Enable Register (LON_MODE)

Name: US_IER (LON_MODE)

Address: 0x40024008 (0), 0x40028008 (1), 0x4002C008 (2)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	LBLOVFE	LRXD	LFET	LCOL	LTXD
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	UNRE	TXEMPTY	–
7	6	5	4	3	2	1	0
LCRCE	LSFE	OVRE	–	–	–	TXRDY	RXRDY

This register is relevant only if USART_MODE = 0x9 in the USART Mode Register.

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Enables the corresponding interrupt.

- **RXRDY: RXRDY Interrupt Enable**
- **TXRDY: TXRDY Interrupt Enable**
- **OVRE: Overrun Error Interrupt Enable**
- **LSFE: LON Short Frame Error Interrupt Enable**
- **LCRCE: LON CRC Error Interrupt Enable**
- **TXEMPTY: TXEMPTY Interrupt Enable**
- **UNRE: Underrun Error Interrupt Enable**
- **LTXD: LON Transmission Done Interrupt Enable**
- **LCOL: LON Collision Interrupt Enable**
- **LFET: LON Frame Early Termination Interrupt Enable**
- **LRXD: LON Reception Done Interrupt Enable**
- **LBLOVFE: LON Backlog Overflow Error Interrupt Enable**

Downloaded from Arrow.com.

45.7.9 USART Interrupt Disable Register

Name: US_IDR

Address: 0x4002400C (0), 0x4002800C (1), 0x4002C00C (2)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	MANE
23	22	21	20	19	18	17	16
–	–	–	–	CTSIC	DCDIC	DSRIC	RIIC
15	14	13	12	11	10	9	8
–	–	NACK	–	–	ITER	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	RXBRK	TXRDY	RXRDY

For SPI specific configuration, see Section 45.7.10 ”USART Interrupt Disable Register (SPI_MODE)”.

For LIN specific configuration, see Section 45.7.11 ”USART Interrupt Disable Register (LIN_MODE)”.

For LON specific configuration, see Section 45.7.12 ”USART Interrupt Disable Register (LON_MODE)”.

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Disables the corresponding interrupt.

- **RXRDY: RXRDY Interrupt Disable**
- **TXRDY: TXRDY Interrupt Disable**
- **RXBRK: Receiver Break Interrupt Disable**
- **OVRE: Overrun Error Interrupt Enable**
- **FRAME: Framing Error Interrupt Disable**
- **PARE: Parity Error Interrupt Disable**
- **TIMEOUT: Time-out Interrupt Disable**
- **TXEMPTY: TXEMPTY Interrupt Disable**
- **ITER: Max Number of Repetitions Reached Interrupt Disable**
- **NACK: Non Acknowledge Interrupt Disable**
- **RIIC: Ring Indicator Input Change Disable**
- **DSRIC: Data Set Ready Input Change Disable**
- **DCDIC: Data Carrier Detect Input Change Interrupt Disable**
- **CTSIC: Clear to Send Input Change Interrupt Disable**

Downloaded from Arrow.com.

- **MANE: Manchester Error Interrupt Disable**

Downloaded from Arrow.com.

45.7.10 USART Interrupt Disable Register (SPI_MODE)

Name: US_IDR (SPI_MODE)

Address: 0x4002400C (0), 0x4002800C (1), 0x4002C00C (2)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	NSSE	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	UNRE	TXEMPTY	–
7	6	5	4	3	2	1	0
–	–	OVRE	–	–	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE = 0xE or 0xF in the USART Mode Register.

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Disables the corresponding interrupt.

- **RXRDY: RXRDY Interrupt Disable**
- **TXRDY: TXRDY Interrupt Disable**
- **OVRE: Overrun Error Interrupt Disable**
- **TXEMPTY: TXEMPTY Interrupt Disable**
- **UNRE: SPI Underrun Error Interrupt Disable**
- **NSSE: NSS Line (Driving CTS Pin) Rising or Falling Edge Event Interrupt Disable**

Downloaded from Arrow.com.

45.7.11 USART Interrupt Disable Register (LIN_MODE)

Name: US_IDR (LIN_MODE)

Address: 0x4002400C (0), 0x4002800C (1), 0x4002C00C (2)

Access: Write-only

31	30	29	28	27	26	25	24
LINHTE	LINSTE	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
LINTC	LINID	LINBK	–	–	–	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE = 0xA or 0xB in the USART Mode Register.

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Disables the corresponding interrupt.

- **RXRDY: RXRDY Interrupt Disable**
- **TXRDY: TXRDY Interrupt Disable**
- **OVRE: Overrun Error Interrupt Disable**
- **FRAME: Framing Error Interrupt Disable**
- **PARE: Parity Error Interrupt Disable**
- **TIMEOUT: Time-out Interrupt Disable**
- **TXEMPTY: TXEMPTY Interrupt Disable**
- **LINBK: LIN Break Sent or LIN Break Received Interrupt Disable**
- **LINID: LIN Identifier Sent or LIN Identifier Received Interrupt Disable**
- **LINTC: LIN Transfer Completed Interrupt Disable**
- **LINBE: LIN Bus Error Interrupt Disable**
- **LINISFE: LIN Inconsistent Synch Field Error Interrupt Disable**
- **LINIPE: LIN Identifier Parity Interrupt Disable**
- **LINCE: LIN Checksum Error Interrupt Disable**
- **LINSNRE: LIN Slave Not Responding Error Interrupt Disable**

Downloaded from Arrow.com.

- **LINSTE: LIN Synch Tolerance Error Interrupt Disable**
- **LINHTE: LIN Header Timeout Error Interrupt Disable**

Downloaded from Arrow.com.

45.7.12 USART Interrupt Disable Register (LON_MODE)

Name: US_IDR (LON_MODE)

Address: 0x4002400C (0), 0x4002800C (1), 0x4002C00C (2)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	LBLOVFE	LRXD	LFET	LCOL	LTXD
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	UNRE	TXEMPTY	–
7	6	5	4	3	2	1	0
LCRCE	LSFE	OVRE	–	–	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE = 0x9 in the USART Mode Register.

The following configuration values are valid for all listed bit names of this register:

0: No effect

1: Disables the corresponding interrupt.

- **RXRDY: RXRDY Interrupt Disable**
- **TXRDY: TXRDY Interrupt Disable**
- **OVRE: Overrun Error Interrupt Disable**
- **LSFE: LON Short Frame Error Interrupt Disable**
- **LCRCE: LON CRC Error Interrupt Disable**
- **TXEMPTY: TXEMPTY Interrupt Disable**
- **UNRE: Underrun Error Interrupt Disable**
- **LTXD: LON Transmission Done Interrupt Disable**
- **LCOL: LON Collision Interrupt Disable**
- **LFET: LON Frame Early Termination Interrupt Disable**
- **LRXD: LON Reception Done Interrupt Disable**
- **LBLOVFE: LON Backlog Overflow Error Interrupt Disable**

Downloaded from Arrow.com.

45.7.13 USART Interrupt Mask Register

Name: US_IMR

Address: 0x40024010 (0), 0x40028010 (1), 0x4002C010 (2)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	MANE
23	22	21	20	19	18	17	16
–	–	–	–	CTSIC	DCDIC	DSRIC	RIIC
15	14	13	12	11	10	9	8
–	–	NACK	–	–	ITER	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	RXBRK	TXRDY	RXRDY

For SPI specific configuration, see Section 45.7.14 ”USART Interrupt Mask Register (SPI_MODE)”.

For LIN specific configuration, see Section 45.7.15 ”USART Interrupt Mask Register (LIN_MODE)”.

For LON specific configuration, see Section 45.7.16 ”USART Interrupt Mask Register (LON_MODE)”.

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

- **RXRDY: RXRDY Interrupt Mask**
- **TXRDY: TXRDY Interrupt Mask**
- **RXBRK: Receiver Break Interrupt Mask**
- **OVRE: Overrun Error Interrupt Mask**
- **FRAME: Framing Error Interrupt Mask**
- **PARE: Parity Error Interrupt Mask**
- **TIMEOUT: Time-out Interrupt Mask**
- **TXEMPTY: TXEMPTY Interrupt Mask**
- **ITER: Max Number of Repetitions Reached Interrupt Mask**
- **NACK: Non Acknowledge Interrupt Mask**
- **RIIC: Ring Indicator Input Change Mask**
- **DSRIC: Data Set Ready Input Change Mask**
- **DCDIC: Data Carrier Detect Input Change Interrupt Mask**
- **CTSIC: Clear to Send Input Change Interrupt Mask**

Downloaded from Arrow.com.

- **MANE: Manchester Error Interrupt Mask**

Downloaded from Arrow.com.

45.7.14 USART Interrupt Mask Register (SPI_MODE)

Name: US_IMR (SPI_MODE)

Address: 0x40024010 (0), 0x40028010 (1), 0x4002C010 (2)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	NSSE	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	UNRE	TXEMPTY	–
7	6	5	4	3	2	1	0
–	–	OVRE	–	–	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE = 0xE or 0xF in the USART Mode Register.

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

- **RXRDY: RXRDY Interrupt Mask**
- **TXRDY: TXRDY Interrupt Mask**
- **OVRE: Overrun Error Interrupt Mask**
- **TXEMPTY: TXEMPTY Interrupt Mask**
- **UNRE: SPI Underrun Error Interrupt Mask**
- **NSSE: NSS Line (Driving CTS Pin) Rising or Falling Edge Event Interrupt Mask**

Downloaded from Arrow.com.

45.7.15 USART Interrupt Mask Register (LIN_MODE)

Name: US_IMR (LIN_MODE)

Address: 0x40024010 (0), 0x40028010 (1), 0x4002C010 (2)

Access: Read-only

31	30	29	28	27	26	25	24
LINHTE	LINSTE	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
LINTC	LINID	LINBK	–	–	–	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE = 0xA or 0xB in the USART Mode Register.

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

- **RXRDY: RXRDY Interrupt Mask**
- **TXRDY: TXRDY Interrupt Mask**
- **OVRE: Overrun Error Interrupt Mask**
- **FRAME: Framing Error Interrupt Mask**
- **PARE: Parity Error Interrupt Mask**
- **TIMEOUT: Time-out Interrupt Mask**
- **TXEMPTY: TXEMPTY Interrupt Mask**
- **LINBK: LIN Break Sent or LIN Break Received Interrupt Mask**
- **LINID: LIN Identifier Sent or LIN Identifier Received Interrupt Mask**
- **LINTC: LIN Transfer Completed Interrupt Mask**
- **LINBE: LIN Bus Error Interrupt Mask**
- **LINISFE: LIN Inconsistent Synch Field Error Interrupt Mask**
- **LINIPE: LIN Identifier Parity Interrupt Mask**
- **LINCE: LIN Checksum Error Interrupt Mask**
- **LINSNRE: LIN Slave Not Responding Error Interrupt Mask**

Downloaded from Arrow.com.

- **LINSTE: LIN Synch Tolerance Error Interrupt Mask**
- **LINHTE: LIN Header Timeout Error Interrupt Mask**

Downloaded from Arrow.com.

45.7.16 USART Interrupt Mask Register (LON_MODE)

Name: US_IMR (LON_MODE)

Address: 0x40024010 (0), 0x40028010 (1), 0x4002C010 (2)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	LBLOVFE	LRXD	LFET	LCOL	LTXD
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	UNRE	TXEMPTY	–
7	6	5	4	3	2	1	0
LCRCE	LSFE	OVRE	–	–	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE = 0x9 in the USART Mode Register.

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

- **RXRDY: RXRDY Interrupt Mask**
- **TXRDY: TXRDY Interrupt Mask**
- **OVRE: Overrun Error Interrupt Mask**
- **LSFE: LON Short Frame Error Interrupt Mask**
- **LCRCE: LON CRC Error Interrupt Mask**
- **TXEMPTY: TXEMPTY Interrupt Mask**
- **UNRE: Underrun Error Interrupt Mask**
- **LTXD: LON Transmission Done Interrupt Mask**
- **LCOL: LON Collision Interrupt Mask**
- **LFET: LON Frame Early Termination Interrupt Mask**
- **LRXD: LON Reception Done Interrupt Mask**
- **LBLOVFE: LON Backlog Overflow Error Interrupt Mask**

Downloaded from Arrow.com.

45.7.17 USART Channel Status Register

Name: US_CSR

Address: 0x40024014 (0), 0x40028014 (1), 0x4002C014 (2)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	MANERR
23	22	21	20	19	18	17	16
CTS	DCD	DSR	RI	CTSIC	DCDIC	DSRIC	RIIC
15	14	13	12	11	10	9	8
–	–	NACK	–	–	ITER	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	RXBRK	TXRDY	RXRDY

For SPI specific configuration, see Section 45.7.18 ”USART Channel Status Register (SPI_MODE)”.

For LIN specific configuration, see Section 45.7.19 ”USART Channel Status Register (LIN_MODE)”.

For LON specific configuration, see Section 45.7.20 ”USART Channel Status Register (LON_MODE)”.

- **RXRDY: Receiver Ready (cleared by reading US_RHR)**

0: No complete character has been received since the last read of US_RHR or the receiver is disabled. If characters were being received when the receiver was disabled, RXRDY changes to 1 when the receiver is enabled.

1: At least one complete character has been received and US_RHR has not yet been read.

- **TXRDY: Transmitter Ready (cleared by writing US_THR)**

0: A character is in the US_THR waiting to be transferred to the Transmit Shift Register, or an STTBRK command has been requested, or the transmitter is disabled. As soon as the transmitter is enabled, TXRDY becomes 1.

1: There is no character in the US_THR.

- **RXBRK: Break Received/End of Break (cleared by writing a one to bit US_CR.RSTSTA)**

0: No break received or end of break detected since the last RSTSTA.

1: Break received or end of break detected since the last RSTSTA.

- **OVRE: Overrun Error (cleared by writing a one to bit US_CR.RSTSTA)**

0: No overrun error has occurred since the last RSTSTA.

1: At least one overrun error has occurred since the last RSTSTA.

- **FRAME: Framing Error (cleared by writing a one to bit US_CR.RSTSTA)**

0: No stop bit has been detected low since the last RSTSTA.

1: At least one stop bit has been detected low since the last RSTSTA.

- **PARE: Parity Error (cleared by writing a one to bit US_CR.RSTSTA)**

0: No parity error has been detected since the last RSTSTA.

1: At least one parity error has been detected since the last RSTSTA.

Downloaded from Arrow.com.

- **TIMEOUT: Receiver Time-out (cleared by writing a one to bit US_CR.STTTO)**

0: There has not been a time-out since the last Start Time-out command (STTTO in US_CR) or the Time-out Register is 0.

1: There has been a time-out since the last Start Time-out command (STTTO in US_CR).

- **TXEMPTY: Transmitter Empty (cleared by writing US_THR)**

0: There are characters in either US_THR or the Transmit Shift Register, or the transmitter is disabled.

1: There are no characters in US_THR, nor in the Transmit Shift Register.

- **ITER: Max Number of Repetitions Reached (cleared by writing a one to bit US_CR.RSTIT)**

0: Maximum number of repetitions has not been reached since the last RSTIT.

1: Maximum number of repetitions has been reached since the last RSTIT.

- **NACK: Non Acknowledge Interrupt (cleared by writing a one to bit US_CR.RSTNACK)**

0: Non acknowledge has not been detected since the last RSTNACK.

1: At least one non acknowledge has been detected since the last RSTNACK.

- **RIIC: Ring Indicator Input Change Flag (cleared on read)**

0: No input change has been detected on the RI pin since the last read of US_CSR.

1: At least one input change has been detected on the RI pin since the last read of US_CSR.

- **DSRIC: Data Set Ready Input Change Flag (cleared on read)**

0: No input change has been detected on the DSR pin since the last read of US_CSR.

1: At least one input change has been detected on the DSR pin since the last read of US_CSR.

- **DCDIC: Data Carrier Detect Input Change Flag (cleared on read)**

0: No input change has been detected on the DCD pin since the last read of US_CSR.

1: At least one input change has been detected on the DCD pin since the last read of US_CSR.

- **CTSIC: Clear to Send Input Change Flag (cleared on read)**

0: No input change has been detected on the CTS pin since the last read of US_CSR.

1: At least one input change has been detected on the CTS pin since the last read of US_CSR.

- **RI: Image of RI Input**

0: RI input is driven low.

1: RI input is driven high.

- **DSR: Image of DSR Input**

0: DSR input is driven low.

1: DSR input is driven high.

- **DCD: Image of DCD Input**

0: DCD input is driven low.

1: DCD input is driven high.

Downloaded from Arrow.com.

- **CTS: Image of CTS Input**

0: CTS input is driven low.

1: CTS input is driven high.

- **MANERR: Manchester Error (cleared by writing a one to the bit US_CR.RSTSTA)**

0: No Manchester error has been detected since the last RSTSTA.

1: At least one Manchester error has been detected since the last RSTSTA.

Downloaded from Arrow.com.

45.7.18 USART Channel Status Register (SPI_MODE)

Name: US_CSR (SPI_MODE)

Address: 0x40024014 (0), 0x40028014 (1), 0x4002C014 (2)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
NSS	–	–	–	NSSE	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	UNRE	TXEMPTY	–
7	6	5	4	3	2	1	0
–	–	OVRE	–	–	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE = 0xE or 0xF in the USART Mode Register.

- **RXRDY: Receiver Ready (cleared by reading US_RHR)**

0: No complete character has been received since the last read of US_RHR or the receiver is disabled. If characters were being received when the receiver was disabled, RXRDY changes to 1 when the receiver is enabled.

1: At least one complete character has been received and US_RHR has not yet been read.

- **TXRDY: Transmitter Ready (cleared by writing US_THR)**

0: A character is in the US_THR waiting to be transferred to the Transmit Shift Register or the transmitter is disabled. As soon as the transmitter is enabled, TXRDY becomes 1.

1: There is no character in the US_THR.

- **OVRE: Overrun Error (cleared by writing a one to bit US_CR.RSTSTA)**

0: No overrun error has occurred since the last RSTSTA.

1: At least one overrun error has occurred since the last RSTSTA.

- **TXEMPTY: Transmitter Empty (cleared by writing US_THR)**

0: There are characters in either US_THR or the Transmit Shift Register, or the transmitter is disabled.

1: There are no characters in US_THR, nor in the Transmit Shift Register.

- **UNRE: Underrun Error (cleared by writing a one to bit US_CR.RSTSTA)**

0: No SPI underrun error has occurred since the last RSTSTA.

1: At least one SPI underrun error has occurred since the last RSTSTA.

- **NSSE: NSS Line (Driving CTS Pin) Rising or Falling Edge Event (cleared on read)**

0: No NSS line event has been detected since the last read of US_CSR.

1: A rising or falling edge event has been detected on NSS line since the last read of US_CSR .

Downloaded from Arrow.com.

- **NSS: Image of NSS Line**

0: NSS line is driven low (if NSSE = 1, falling edge occurred on NSS line).

1: NSS line is driven high (if NSSE = 1, rising edge occurred on NSS line).

Downloaded from Arrow.com.

45.7.19 USART Channel Status Register (LIN_MODE)

Name: US_CSR (LIN_MODE)

Address: 0x40024014 (0), 0x40028014 (1), 0x4002C014 (2)

Access: Read-only

31	30	29	28	27	26	25	24
LINHTE	LINSTE	LINSNRE	LINCE	LINIPE	LINISFE	LINBE	–
23	22	21	20	19	18	17	16
LINBLS	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
LINTC	LINID	LINBK	–	–	–	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE = 0xA or 0xB in the USART Mode Register.

- **RXRDY: Receiver Ready (cleared by reading US_THR)**

0: No complete character has been received since the last read of US_RHR or the receiver is disabled. If characters were being received when the receiver was disabled, RXRDY changes to 1 when the receiver is enabled.

1: At least one complete character has been received and US_RHR has not yet been read.

- **TXRDY: Transmitter Ready (cleared by writing US_THR)**

0: A character is in the US_THR waiting to be transferred to the Transmit Shift Register or the transmitter is disabled. As soon as the transmitter is enabled, TXRDY becomes 1.

1: There is no character in the US_THR.

- **OVRE: Overrun Error (cleared by writing a one to bit US_CR.RSTSTA)**

0: No overrun error has occurred since the last RSTSTA.

1: At least one overrun error has occurred since the last RSTSTA.

- **FRAME: Framing Error (cleared by writing a one to bit US_CR.RSTSTA)**

0: No stop bit has been detected low since the last RSTSTA.

1: At least one stop bit has been detected low since the last RSTSTA.

- **PARE: Parity Error (cleared by writing a one to bit US_CR.RSTSTA)**

0: No parity error has been detected since the last RSTSTA.

1: At least one parity error has been detected since the last RSTSTA.

- **TIMEOUT: Receiver Time-out (cleared by writing a one to bit US_CR.RSTSTA)**

0: There has not been a time-out since the last start time-out command (STTTO in US_CR) or the Time-out Register is 0.

1: There has been a time-out since the last start time-out command (STTTO in US_CR).

Downloaded from Arrow.com.

- **TXEMPTY: Transmitter Empty (cleared by writing US_THR)**

0: There are characters in either US_THR or the Transmit Shift Register, or the transmitter is disabled.

1: There are no characters in US_THR, nor in the Transmit Shift Register.

- **LINBK: LIN Break Sent or LIN Break Received (cleared by writing a one to bit US_CR.RSTSTA)**

Applicable if USART operates in LIN master mode (USART_MODE = 0xA):

0: No LIN break has been sent since the last RSTSTA.

1:At least one LIN break has been sent since the last RSTSTA

If USART operates in LIN Slave mode (USART_MODE = 0xB):

0: No LIN break has received sent since the last RSTSTA.

1:At least one LIN break has been received since the last RSTSTA.

- **LINID: LIN Identifier Sent or LIN Identifier Received (cleared by writing a one to bit US_CR.RSTSTA)**

If USART operates in LIN Master mode (USART_MODE = 0xA):

0: No LIN identifier has been sent since the last RSTSTA.

1:At least one LIN identifier has been sent since the last RSTSTA.

If USART operates in LIN Slave mode (USART_MODE = 0xB):

0: No LIN identifier has been received since the last RSTSTA.

1:At least one LIN identifier has been received since the last RSTSTA

- **LINTC: LIN Transfer Completed (cleared by writing a one to bit US_CR.RSTSTA)**

0: The USART is idle or a LIN transfer is ongoing.

1: A LIN transfer has been completed since the last RSTSTA.

- **LINBLS: LIN Bus Line Status**

0: LIN bus line is set to 0.

1: LIN bus line is set to 1.

- **LINBE: LIN Bit Error (cleared by writing a one to bit US_CR.RSTSTA)**

0: No bit error has been detected since the last RSTSTA.

1: A bit error has been detected since the last RSTSTA.

- **LINISFE: LIN Inconsistent Synch Field Error (cleared by writing a one to bit US_CR.RSTSTA)**

0: No LIN inconsistent synch field error has been detected since the last RSTSTA

1: The USART is configured as a slave node and a LIN Inconsistent synch field error has been detected since the last RSTSTA.

- **LINIPE: LIN Identifier Parity Error (cleared by writing a one to bit US_CR.RSTSTA)**

0: No LIN identifier parity error has been detected since the last RSTSTA.

1: A LIN identifier parity error has been detected since the last RSTSTA.

- **LINCE: LIN Checksum Error (cleared by writing a one to bit US_CR.RSTSTA)**

0: No LIN checksum error has been detected since the last RSTSTA.

1: A LIN checksum error has been detected since the last RSTSTA.

Downloaded from Arrow.com.

- **LINSNRE: LIN Slave Not Responding Error (cleared by writing a one to bit US_CR.RSTSTA)**

0: No LIN slave not responding error has been detected since the last RSTSTA.

1: A LIN slave not responding error has been detected since the last RSTSTA.

- **LINSTE: LIN Synch Tolerance Error (cleared by writing a one to bit US_CR.RSTSTA)**

0: No LIN synch tolerance error has been detected since the last RSTSTA.

1: A LIN synch tolerance error has been detected since the last RSTSTA.

- **LINHTE: LIN Header Timeout Error (cleared by writing a one to bit US_CR.RSTSTA)**

0: No LIN header timeout error has been detected since the last RSTSTA.

1: A LIN header timeout error has been detected since the last RSTSTA.

Downloaded from Arrow.com.

45.7.20 USART Channel Status Register (LON_MODE)

Name: US_CSR (LON_MODE)

Address: 0x40024014 (0), 0x40028014 (1), 0x4002C014 (2)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	LBLOVFE	LRXD	LFET	LCOL	LTXD
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	UNRE	TXEMPTY	–
7	6	5	4	3	2	1	0
LCRCE	LSFE	OVRE	–	–	–	TXRDY	RXRDY

This configuration is relevant only if USART_MODE = 0x9 in the USART Mode Register.

- **RXRDY: Receiver Ready (cleared by reading US_RHR)**

0: No complete character has been received since the last read of US_RHR or the receiver is disabled. If characters were being received when the receiver was disabled, RXRDY changes to 1 when the receiver is enabled.

1: At least one complete character has been received and US_RHR has not yet been read.

- **TXRDY: Transmitter Ready (cleared by writing US_THR)**

0: A character is in the US_THR waiting to be transferred to the Transmit Shift Register or the transmitter is disabled. As soon as the transmitter is enabled, TXRDY becomes 1.

1: There is no character in the US_THR.

- **OVRE: Overrun Error (cleared by writing a one to bit US_CR.RSTSTA)**

0: No overrun error has occurred since the last RSTSTA.

1: At least one overrun error has occurred since the last RSTSTA.

- **LSFE: LON Short Frame Error (cleared by writing a one to bit US_CR.RSTSTA)**

0: No short frame received since the last RSTSTA.

1: At least one short frame received since the last RSTSTA.

- **LCRCE: LON CRC Error (cleared by writing a one to bit US_CR.RSTSTA)**

0: No CRC error has been detected since the last RSTSTA.

1: At least one CRC error has been detected since the last RSTSTA.

- **TXEMPTY: Transmitter Empty (cleared by writing US_THR)**

0: There are characters in either US_THR or the Transmit Shift Register, or the transmitter is disabled.

1: There are no characters in US_THR, nor in the Transmit Shift Register.

Downloaded from Arrow.com.

- **UNRE: Underrun Error (cleared by writing a one to bit US_CR.RSTSTA)**

0: No LON underrun error has occurred since the last RSTSTA.

1: At least one LON underrun error has occurred since the last RSTSTA.

- **LTXD: LON Transmission End Flag (cleared by writing a one to bit US_CR.RSTSTA)**

0: Transmission on going or no transmission occurred since the last RSTSTA.

1: At least one transmission has been performed since the last RSTSTA.

- **LCOL: LON Collision Detected Flag (cleared by writing a one to bit US_CR.RSTSTA)**

0: No collision occurred while transmitting since the last RSTSTA.

1: At least one collision occurred while transmitting since the last RSTSTA.

- **LFET: LON Frame Early Termination (cleared by writing a one to bit US_CR.RSTSTA)**

0: No frame has been terminated early due to collision detection since the last RSTSTA.

1: At least one transmission has been terminated due to collision detection since the last RSTSTA. (This stops the DMA until reset with RSTSTA bit).

- **LRXD: LON Reception End Flag (cleared by writing a one to bit US_CR.RSTSTA)**

0: Reception on going or no reception occurred since the last RSTSTA.

1: At least one reception has been performed since the last RSTSTA.

- **LBLOVFE: LON Backlog Overflow Error (cleared by writing a one to bit US_CR.RSTSTA)**

0: No backlog overflow error occurred since the last RSTSTA.

1: At least one backlog error overflow occurred since the last RSTSTA.

Downloaded from Arrow.com.

45.7.21 USART Receive Holding Register

Name: US_RHR

Address: 0x40024018 (0), 0x40028018 (1), 0x4002C018 (2)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
RXSYNH	–	–	–	–	–	–	RXCHR

7	6	5	4	3	2	1	0
RXCHR							

- **RXCHR: Received Character**

Last character received if RXRDY is set.

- **RXSYNH: Received Sync**

0: Last character received is a data.

1: Last character received is a command.

Downloaded from Arrow.com.

45.7.22 USART Transmit Holding Register

Name: US_THR

Address: 0x4002401C (0), 0x4002801C (1), 0x4002C01C (2)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
TXSYNH	–	–	–	–	–	–	TXCHR

7	6	5	4	3	2	1	0
TXCHR							

- **TXCHR: Character to be Transmitted**

Next character to be transmitted after the current character if TXRDY is not set.

- **TXSYNH: Sync Field to be Transmitted**

0: The next character sent is encoded as a data. Start frame delimiter is DATA SYNC.

1: The next character sent is encoded as a command. Start frame delimiter is COMMAND SYNC.

Downloaded from Arrow.com.

45.7.23 USART Baud Rate Generator Register

Name: US_BRGR

Address: 0x40024020 (0), 0x40028020 (1), 0x4002C020 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	FP		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CD							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CD							

This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

- **CD: Clock Divider**

	USART_MODE ≠ ISO7816			
	SYNC = 0		SYNC = 1 or USART_MODE = SPI (Master or Slave)	
CD	**OVER = 0**	**OVER = 1**		**USART_MODE = ISO7816**
0	Baud Rate Clock Disabled			
1 to 65535	CD = Selected Clock / (16 × Baud Rate)	CD = Selected Clock / (8 × Baud Rate)	CD = Selected Clock / Baud Rate	CD = Selected Clock / (FI_DI_RATIO × Baud Rate)

- **FP: Fractional Part**

0: Fractional divider is disabled.

1–7: Baud rate resolution, defined by FP × 1/8.

Warning: When the value of field FP is greater than 0, the SCK (oversampling clock) generates non-constant duty cycles. The SCK high duration is increased by “selected clock” period from time to time. The duty cycle depends on the value of the CD field.

Downloaded from Arrow.com.

45.7.24 USART Receiver Time-out Register

Name: US_RTOR

Address: 0x40024024 (0), 0x40028024 (1), 0x4002C024 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	TO

15	14	13	12	11	10	9	8
TO							

7	6	5	4	3	2	1	0
TO							

This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

- **TO: Time-out Value**

0: The receiver time-out is disabled.

1–131071: The receiver time-out is enabled and TO is Time-out Delay / Bit Period.

Downloaded from Arrow.com.

45.7.25 USART Transmitter Timeguard Register

Name: US_TTGR

Address: 0x40024028 (0), 0x40028028 (1), 0x4002C028 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
TG							

This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

For LON specific configuration, see Section 45.7.26 ”USART Transmitter Timeguard Register (LON_MODE)”.

- **TG: Timeguard Value**

0: The transmitter timeguard is disabled.

1–255: The transmitter timeguard is enabled and TG is Timeguard Delay / Bit Period.

Downloaded from Arrow.com.

45.7.26 USART Transmitter Timeguard Register (LON_MODE)

Name: US_TTGR (LON_MODE)

Address: 0x40024028 (0), 0x40028028 (1), 0x4002C028 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
PCYCLE							

15	14	13	12	11	10	9	8
PCYCLE							

7	6	5	4	3	2	1	0
PCYCLE							

This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

- **PCYCLE: LON PCYCLE Length**

1–16777215: LON PCYCLE length in t_{bit}.

Downloaded from Arrow.com.

45.7.27 USART FI DI RATIO Register

Name: US_FIDI

Address: 0x40024040 (0), 0x40028040 (1), 0x4002C040 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
FI_DI_RATIO							
7	6	5	4	3	2	1	0
FI_DI_RATIO							

This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

For LON specific configuration, see Section 45.7.26 ”USART Transmitter Timeguard Register (LON_MODE)”.

- **FI_DI_RATIO: FI Over DI Ratio Value**

0: If ISO7816 mode is selected, the baud rate generator generates no signal.

1–2: Do not use.

3–2047: If ISO7816 mode is selected, the baud rate is the clock provided on SCK divided by FI_DI_RATIO.

Downloaded from Arrow.com.

45.7.28 USART FI DI RATIO Register (LON_MODE)

Name: US_FIDI (LON_MODE)

Address: 0x40024040 (0), 0x40028040 (1), 0x4002C040 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
BETA2							

15	14	13	12	11	10	9	8
BETA2							

7	6	5	4	3	2	1	0
BETA2							

This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

- **BETA2: LON BETA2 Length**

1–16777215: LON BETA2 length in t_{bit}.

Downloaded from Arrow.com.

45.7.29 USART Number of Errors Register

Name: US_NER

Address: 0x40024044 (0), 0x40028044 (1), 0x4002C044 (2)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
NB_ERRORS							

This register is relevant only if USART_MODE = 0x4 or 0x6 in the USART Mode Register.

- **NB_ERRORS: Number of Errors**

Total number of errors that occurred during an ISO7816 transfer. This register automatically clears when read.

Downloaded from Arrow.com.

45.7.30 USART IrDA Filter Register

Name: US_IF

Address: 0x4002404C (0), 0x4002804C (1), 0x4002C04C (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
IRDA_FILTER							

This register is relevant only if USART_MODE = 0x8 in the USART Mode Register.
This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

- **IRDA_FILTER: IrDA Filter**

The IRDA_FILTER value must be defined to meet the following criteria:

$t_{peripheral\ clock} \times (IRDA_FILTER + 3) < 1.41\ \mu s$

Downloaded from Arrow.com.

45.7.31 USART Manchester Configuration Register

Name: US_MAN

Address: 0x40024050 (0), 0x40028050 (1), 0x4002C050 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
RXIDLEV	DRIFT	ONE	RX_MPOL	–	–	RX_PP	

23	22	21	20	19	18	17	16
–	–	–	–	RX_PL			

15	14	13	12	11	10	9	8
–	–	–	TX_MPOL	–	–	TX_PP	

7	6	5	4	3	2	1	0
–	–	–	–	TX_PL			

This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

- **TX_PL: Transmitter Preamble Length**

0: The transmitter preamble pattern generation is disabled

1–15: The preamble length is TX_PL × Bit Period

- **TX_PP: Transmitter Preamble Pattern**

The following values assume that TX_MPOL field is not set:

Value	Name	Description
0	ALL_ONE	The preamble is composed of ‘1’s
1	ALL_ZERO	The preamble is composed of ‘0’s
2	ZERO_ONE	The preamble is composed of ‘01’s
3	ONE_ZERO	The preamble is composed of ‘10’s

- **TX_MPOL: Transmitter Manchester Polarity**

0: Logic zero is coded as a zero-to-one transition, Logic one is coded as a one-to-zero transition.

1: Logic zero is coded as a one-to-zero transition, Logic one is coded as a zero-to-one transition.

- **RX_PL: Receiver Preamble Length**

0: The receiver preamble pattern detection is disabled

1–15: The detected preamble length is RX_PL × Bit Period

Downloaded from Arrow.com.

- **RX_PP: Receiver Preamble Pattern detected**

The following values assume that RX_MPOL field is not set:

Value	Name	Description
00	ALL_ONE	The preamble is composed of ‘1’s
01	ALL_ZERO	The preamble is composed of ‘0’s
10	ZERO_ONE	The preamble is composed of ‘01’s
11	ONE_ZERO	The preamble is composed of ‘10’s

- **RX_MPOL: Receiver Manchester Polarity**

0: Logic zero is coded as a zero-to-one transition, Logic one is coded as a one-to-zero transition.

1: Logic zero is coded as a one-to-zero transition, Logic one is coded as a zero-to-one transition.

- **ONE: Must Be Set to 1**

Bit 29 must always be set to 1 when programming the US_MAN register.

- **DRIFT: Drift Compensation**

0: The USART cannot recover from an important clock drift

1: The USART can recover from clock drift. The 16X clock mode must be enabled.

- **RXIDLEV: Receiver Idle Value**

0: Receiver line idle value is 0.

1: Receiver line idle value is 1.

Downloaded from Arrow.com.

45.7.32 USART LIN Mode Register

Name: US_LINMR

Address: 0x40024054 (0), 0x40028054 (1), 0x4002C054 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	SYNCDIS	PDCM
15	14	13	12	11	10	9	8
DLC							
7	6	5	4	3	2	1	0
WKUPTYP	FSDIS	DLM	CHKTYP	CHKDIS	PARDIS	NACT	

This register is relevant only if USART_MODE = 0xA or 0xB in the USART Mode Register.
This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

- **NACT: LIN Node Action**

Value	Name	Description
00	PUBLISH	The USART transmits the response.
01	SUBSCRIBE	The USART receives the response.
10	IGNORE	The USART does not transmit and does not receive the response.

Values which are not listed in the table must be considered as “reserved”.

- **PARDIS: Parity Disable**

0: In master node configuration, the identifier parity is computed and sent automatically. In master node and slave node configuration, the parity is checked automatically.

1: Whatever the node configuration is, the Identifier parity is not computed/sent and it is not checked.

- **CHKDIS: Checksum Disable**

0: In master node configuration, the checksum is computed and sent automatically. In slave node configuration, the checksum is checked automatically.

1: Whatever the node configuration is, the checksum is not computed/sent and it is not checked.

- **CHKTYP: Checksum Type**

0: LIN 2.0 “enhanced” checksum

1: LIN 1.3 “classic” checksum

- **DLM: Data Length Mode**

0: The response data length is defined by field DLC of this register.

1: The response data length is defined by bits 5 and 6 of the identifier (IDCHR in US_LINIR).

Downloaded from Arrow.com.

- **FSDIS: Frame Slot Mode Disable**

0: The Frame Slot mode is enabled.

1: The Frame Slot mode is disabled.

- **WKUPTYP: Wakeup Signal Type**

0: Setting the bit LINWKUP in the control register sends a LIN 2.0 wakeup signal.

1: Setting the bit LINWKUP in the control register sends a LIN 1.3 wakeup signal.

- **DLC: Data Length Control**

0–255: Defines the response data length if DLM = 0,in that case the response data length is equal to DLC+1 bytes.

- **PDCM: DMAC Mode**

0: The LIN mode register US_LINMR is not written by the DMAC.

1: The LIN mode register US_LINMR (excepting that flag) is written by the DMAC.

- **SYNCDIS: Synchronization Disable**

0: The synchronization procedure is performed in LIN slave node configuration.

1: The synchronization procedure is not performed in LIN slave node configuration.

Downloaded from Arrow.com.

45.7.33 USART LIN Identifier Register

Name: US_LINIR

Address: 0x40024058 (0), 0x40028058 (1), 0x4002C058 (2)

Access: Read/Write or Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
IDCHR							

This register is relevant only if USART_MODE = 0xA or 0xB in the USART Mode Register.

- **IDCHR: Identifier Character**

If USART_MODE = 0xA (master node configuration):

IDCHR is Read/Write and its value is the identifier character to be transmitted.

If USART_MODE = 0xB (slave node configuration):

IDCHR is Read-only and its value is the last identifier character that has been received.

Downloaded from Arrow.com.

45.7.34 USART LIN Baud Rate Register

Name: US_LINBRR

Address: 0x4002405C (0), 0x4002805C (1), 0x4002C05C (2)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	LINFP		

15	14	13	12	11	10	9	8
LINCD							

7	6	5	4	3	2	1	0
LINCD							

This register is relevant only if USART_MODE = 0xA or 0xB in the USART Mode Register.

Returns the baud rate value after the synchronization process completion.

- **LINCD: Clock Divider after Synchronization**
- **LINFP: Fractional Part after Synchronization**

Downloaded from Arrow.com.

45.7.35 USART LON Mode Register

Name: US_LONMR

Address: 0x40024060 (0), 0x40028060 (1), 0x4002C060 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
EOFS							

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	LCDS	DMAM	CDTAIL	TCOL	COLDET	COMMT

This register is relevant only if USART_MODE = 0x9 in the USART Mode Register.
This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

- **COMMT: LON *comm_type* Parameter Value**

0: LON *comm_type* = *1* mode.

1: LON *comm_type* = *2* mode.

- **COLDET: LON Collision Detection Feature**

0: LON collision detection feature disabled.

1: LON collision detection feature enabled.

- **TCOL: Terminate Frame upon Collision Notification**

0: Do not terminate the frame in LON *comm_type* = *1* mode upon collision detection.

1: Terminate the frame in LON *comm_type* = *1* mode upon collision detection if possible.

- **CDTAIL: LON Collision Detection on Frame Tail**

0: Detect collisions after CRC has been sent but prior end of transmission in LON *comm_type* = *1* mode.

1: Ignore collisions after CRC has been sent but prior end of transmission in LON *comm_type* = *1* mode.

- **DMAM: LON DMA Mode**

0: The LON data length register US_LONDL is not written by the DMA.

1: The LON data length register US_LONDL is written by the DMA.

- **LCDS: LON Collision Detection Source**

0: LON collision detection source is external.

1: LON collision detection source is internal.

- **EOFS: End of Frame Condition Size**

0–255: Define the minimum transitionless time for the IP to detect a LON end of frame condition.
$t_{eof} = (EOFS + 1) \times t_{clock} \times 8 \times (2 - OVER)$

Downloaded from Arrow.com.

45.7.36 USART LON Preamble Register

Name: US_LONPR

Address: 0x40024064 (0), 0x40028064 (1), 0x4002C064 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	LONPL					
7	6	5	4	3	2	1	0
LONPL							

This register is relevant only if USART_MODE = 0x9 in the USART Mode Register.
This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

- **LONPL: LON Preamble Length**

1–16383: LON preamble length in t_{bit} (without byte-sync).

Downloaded from Arrow.com.

45.7.37 USART LON Data Length Register

Name: US_LONDL

Address: 0x40024068 (0), 0x40028068 (1), 0x4002C068 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
LONDL							

This register is relevant only if USART_MODE = 0x9 in the USART Mode Register.

- **LONDL: LON Data Length**

0–255: LON data length is LONDL+1 bytes.

Downloaded from Arrow.com.

45.7.38 USART LON L2HDR Register

Name: US_LONL2HDR

Address: 0x4002406C (0), 0x4002806C (1), 0x4002C06C (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
PB	ALTP	BLI					

This register is relevant only if USART_MODE = 0x9 in the USART Mode Register.

- **BLI: LON Backlog Increment**

0–63: LON backlog increment to be generated as a result of delivering the LON frame.

- **ALTP: LON Alternate Path Bit**

0: LON alternate path bit reset.

1: LON alternate path bit set.

- **PB: LON Priority Bit**

0: LON priority bit reset.

1: LON priority bit set.

Downloaded from Arrow.com.

45.7.39 USART LON Backlog Register

Name: US_LONBL

Address: 0x40024070 (0), 0x40028070 (1), 0x4002C070 (2)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	LONBL					

This register is relevant only if USART_MODE = 0x9 in the USART Mode Register.

- **LONBL: LON Node Backlog Value**

1–63: LON node backlog value.

Downloaded from Arrow.com.

45.7.40 USART LON Beta1 Tx Register

Name: US_LONB1TX

Address: 0x40024074 (0), 0x40028074 (1), 0x4002C074 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
BETA1TX							

15	14	13	12	11	10	9	8
BETA1TX							

7	6	5	4	3	2	1	0
BETA1TX							

This register is relevant only if USART_MODE = 0x9 in the USART Mode Register.
This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

- **BETA1TX: LON Beta1 Length after Transmission**

1–16777215: LON beta1 length after transmission in t_{bit}.

Downloaded from Arrow.com.

45.7.41 USART LON Beta1 Rx Register

Name: US_LONB1RX

Address: 0x40024078 (0), 0x40028078 (1), 0x4002C078 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
BETA1RX							

15	14	13	12	11	10	9	8
BETA1RX							

7	6	5	4	3	2	1	0
BETA1RX							

This register is relevant only if USART_MODE = 0x9 in the USART Mode Register.
This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

- **BETA1RX: LON Beta1 Length after Reception**

1–16777215: LON beta1 length after reception in t_{bit}.

Downloaded from Arrow.com.

45.7.42 USART LON Priority Register

Name: US_LONPRIO

Address: 0x4002407C (0), 0x4002807C (1), 0x4002C07C (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	NPS						

7	6	5	4	3	2	1	0
–	PSNB						

This register is relevant only if USART_MODE = 0x9 in the USART Mode Register.
This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

- **PSNB: LON Priority Slot Number**

0–127: Number of priority slots in the LON network configuration.

- **NPS: LON Node Priority Slot**

0–127: Node priority slot.

Downloaded from Arrow.com.

45.7.43 USART LON IDT Tx Register

Name: US_IDTTX

Address: 0x40024080 (0), 0x40028080 (1), 0x4002C080 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
IDTTX							

15	14	13	12	11	10	9	8
IDTTX							

7	6	5	4	3	2	1	0
IDTTX							

This register is relevant only if USART_MODE = 0x9 in the USART Mode Register.
This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

- **IDTTX: LON Indeterminate Time after Transmission (*comm_type* = *1* mode only)**

0–16777215: LON indeterminate time after transmission in t_{bit}.

Downloaded from Arrow.com.

45.7.44 USART LON IDT Rx Register

Name: US_IDTRX

Address: 0x40024084 (0), 0x40028084 (1), 0x4002C084 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
IDTRX							

15	14	13	12	11	10	9	8
IDTRX							

7	6	5	4	3	2	1	0
IDTRX							

This register is relevant only if USART_MODE = 0x9 in the USART Mode Register.
This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

- **IDTRX: LON Indeterminate Time after Reception (*comm_type = 1* mode only)**

0–16777215: LON indeterminate time after reception in t_{bit}.

Downloaded from Arrow.com.

45.7.45 USART IC DIFF Register

Name: US_ICDIFF

Address: 0x40024088 (0), 0x40028088 (1), 0x4002C088 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	ICDIFF			

This register is relevant only if USART_MODE = 0x9 in the USART Mode Register.
This register can only be written if the WPEN bit is cleared in the USART Write Protection Mode Register.

- **ICDIFF: IC Differentiator Number**

Downloaded from Arrow.com.

45.7.46 USART Write Protection Mode Register

Name: US_WPMR

Address: 0x400240E4 (0), 0x400280E4 (1), 0x4002C0E4 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							
23	22	21	20	19	18	17	16
WPKEY							
15	14	13	12	11	10	9	8
WPKEY							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x555341 (“USA” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x555341 (“USA” in ASCII).

See Section 45.6.12 ”Register Write Protection” for the list of registers that can be write-protected.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x555341	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

Downloaded from Arrow.com.

45.7.47 USART Write Protection Status Register

Name: US_WPSR

Address: 0x400240E8 (0), 0x400280E8 (1), 0x4002C0E8 (2)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

- **WPVS: Write Protection Violation Status**

0: No write protection violation has occurred since the last read of the US_WPSR.

1: A write protection violation has occurred since the last read of the US_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protection Violation Source**

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

Downloaded from Arrow.com.

46. Universal Asynchronous Receiver Transmitter (UART)

46.1 Description

The Universal Asynchronous Receiver Transmitter (UART) features a two-pin UART that can be used for communication and trace purposes and offers an ideal medium for in-situ programming solutions.

Moreover, the association with a DMA controller permits packet handling for these tasks with processor time reduced to a minimum.

46.2 Embedded Characteristics

- Two-pin UART
  - Independent Receiver and Transmitter with a Common Programmable Baud Rate Generator
  - Baud Rate can be Driven by Processor Independent Source Clock
  - Even, Odd, Mark or Space Parity Generation
  - Parity, Framing and Overrun Error Detection
  - Automatic Echo, Local Loopback and Remote Loopback Channel Modes
  - Digital Filter on Receive Line
  - Interrupt Generation
  - Support for Two DMA Channels with Connection to Receiver and Transmitter
  - Supports Asynchronous Partial Wake-up on Receive Line Activity (SleepWalking)
  - Comparison Function on Received Character
  - Register Write Protection

46.3 Block Diagram

Figure 46-1. UART Block Diagram

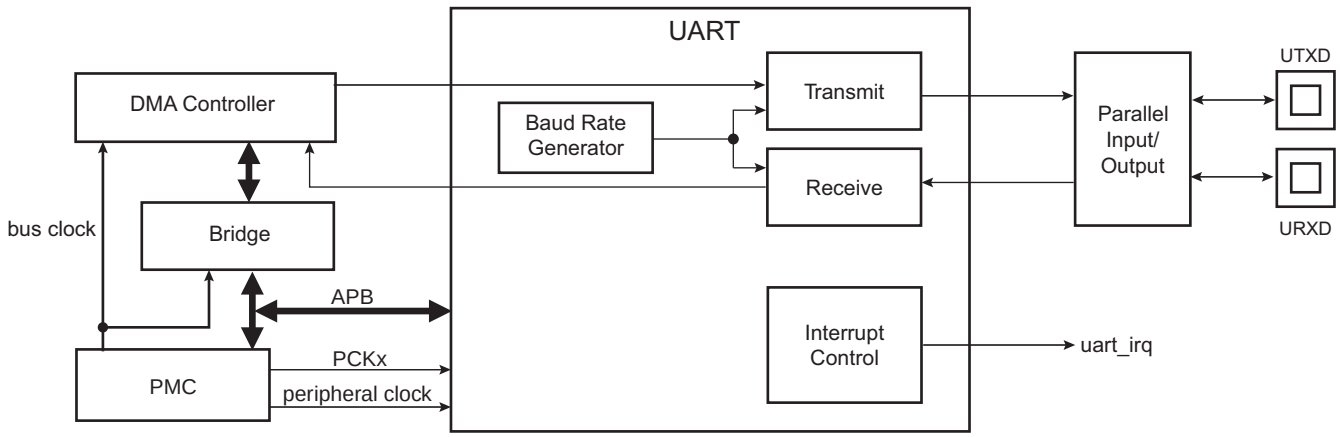


Figure 46-2.

Table 46-1. UART Pin Description

Pin Name	Description	Type
URXD	UART Receive Data	Input
UTXD	UART Transmit Data	Output

Downloaded from Arrow.com.

46.4 Product Dependencies

46.4.1 I/O Lines

The UART pins are multiplexed with PIO lines. The user must first configure the corresponding PIO Controller to enable I/O line operations of the UART.

Table 46-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
UART0	URXD0	PA9	A
UART0	UTXD0	PA10	A
UART1	URXD1	PA5	C
UART1	UTXD1	PA4	C
UART1	UTXD1	PA6	C
UART1	UTXD1	PD26	D
UART2	URXD2	PD25	C
UART2	UTXD2	PD26	C
UART3	URXD3	PD28	A
UART3	UTXD3	PD30	A
UART3	UTXD3	PD31	B
UART4	URXD4	PD18	C
UART4	UTXD4	PD3	C
UART4	UTXD4	PD19	C

46.4.2 Power Management

The UART clock can be controlled through the Power Management Controller (PMC). In this case, the user must first configure the PMC to enable the UART clock. Usually, the peripheral identifier used for this purpose is 1.

In SleepWalking mode (asynchronous partial wake-up), the PMC must be configured to enable SleepWalking for the UART in the Sleepwalking Enable Register (PMC_SLPWK_ER). Depending on the instructions (requests) provided by the UART to the PMC, the system clock may or may not be automatically provided to the UART.

46.4.3 Interrupt Sources

The UART interrupt line is connected to one of the interrupt sources of the Interrupt Controller. Interrupt handling requires programming of the Interrupt Controller before configuring the UART.

Table 46-3. Peripheral IDs

Instance	ID
UART0	7
UART1	8
UART2	44
UART3	45
UART4	46

Downloaded from Arrow.com.

46.5 Functional Description

The UART operates in Asynchronous mode only and supports only 8-bit character handling (with parity). It has no clock pin.

The UART is made up of a receiver and a transmitter that operate independently, and a common baud rate generator. Receiver timeout and transmitter time guard are not implemented. However, all the implemented features are compatible with those of a standard USART.

46.5.1 Baud Rate Generator

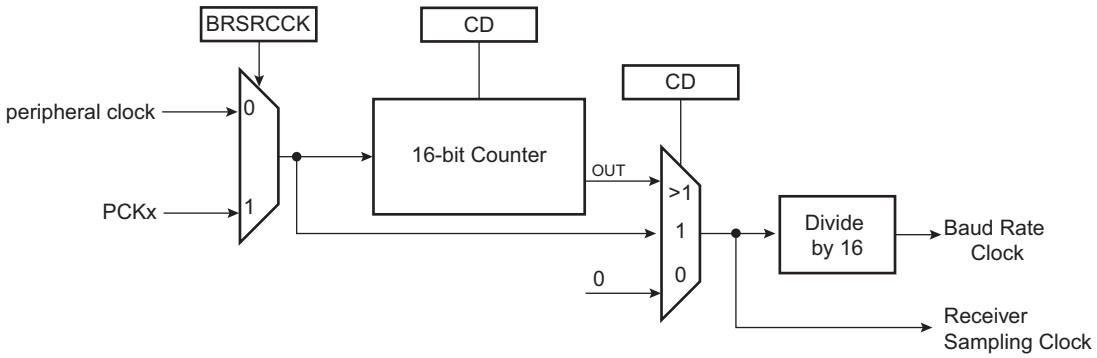
The baud rate generator provides the bit period clock named baud rate clock to both the receiver and the transmitter.

The baud rate clock is the peripheral clock divided by 16 times the clock divisor (CD) value written in the Baud Rate Generator register (UART_BRGR). If UART_BRGR is set to 0, the baud rate clock is disabled and the UART remains inactive. The maximum allowable baud rate is peripheral clock or PMC PCK (PCK) divided by 16. The minimum allowable baud rate is peripheral clock or PCK divided by (16 x 65536). The clock source driving the baud rate generator (peripheral clock or PCK) can be selected by writing the bit BRSRCCK in UART_MR.

If PCK is selected, the baud rate is independent of the processor/bus clock. Thus the processor clock can be changed while UART is enabled. The processor clock frequency changes must be performed only by programming the field PRES in PMC_MCKR (see Section 31. "Power Management Controller (PMC)"). Other methods to modify the processor/bus clock frequency (PLL multiplier, etc.) are forbidden when UART is enabled.

The peripheral clock frequency must be at least three times higher than PCK.

Figure 46-3. Baud Rate Generator



46.5.2 Receiver

46.5.2.1 Receiver Reset, Enable and Disable

After device reset, the UART receiver is disabled and must be enabled before being used. The receiver can be enabled by writing the Control Register (UART_CR) with the bit RXEN at 1. At this command, the receiver starts looking for a start bit.

The programmer can disable the receiver by writing UART_CR with the bit RXDIS at 1. If the receiver is waiting for a start bit, it is immediately stopped. However, if the receiver has already detected a start bit and is receiving the data, it waits for the stop bit before actually stopping its operation.

The receiver can be put in reset state by writing UART_CR with the bit RSTRX at 1. In this case, the receiver immediately stops its current operations and is disabled, whatever its current state. If RSTRX is applied when data is being processed, this data is lost.

Downloaded from Arrow.com.

46.5.2.2 Start Detection and Data Sampling

The UART only supports asynchronous operations, and this affects only its receiver. The UART receiver detects the start of a received character by sampling the URXD signal until it detects a valid start bit. A low level (space) on URXD is interpreted as a valid start bit if it is detected for more than seven cycles of the sampling clock, which is 16 times the baud rate. Hence, a space that is longer than 7/16 of the bit period is detected as a valid start bit. A space which is 7/16 of a bit period or shorter is ignored and the receiver continues to wait for a valid start bit.

When a valid start bit has been detected, the receiver samples the URXD at the theoretical midpoint of each bit. It is assumed that each bit lasts 16 cycles of the sampling clock (1-bit period) so the bit sampling point is eight cycles (0.5-bit period) after the start of the bit. The first sampling point is therefore 24 cycles (1.5-bit periods) after detecting the falling edge of the start bit.

Each subsequent bit is sampled 16 cycles (1-bit period) after the previous one.

Figure 46-4. Start Bit Detection

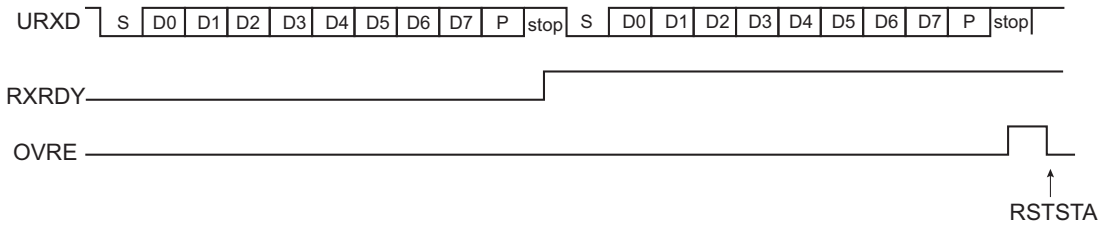
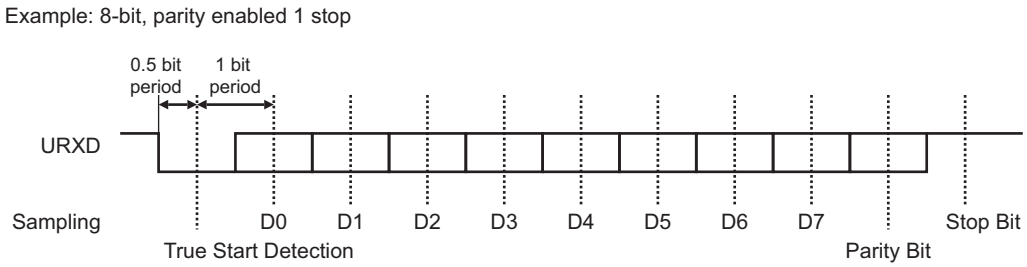


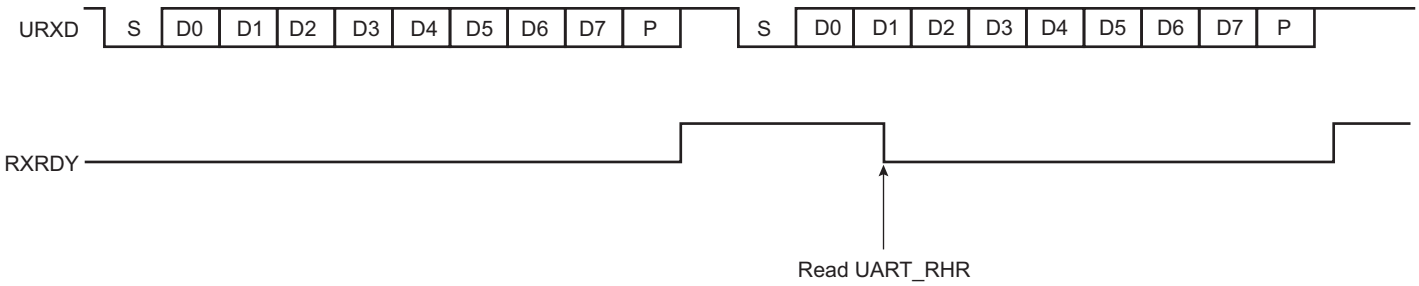
Figure 46-5. Character Reception



46.5.2.3 Receiver Ready

When a complete character is received, it is transferred to the Receive Holding Register (UART_RHR) and the RXRDY status bit in the Status Register (UART_SR) is set. The bit RXRDY is automatically cleared when UART_RHR is read.

Figure 46-6. Receiver Ready

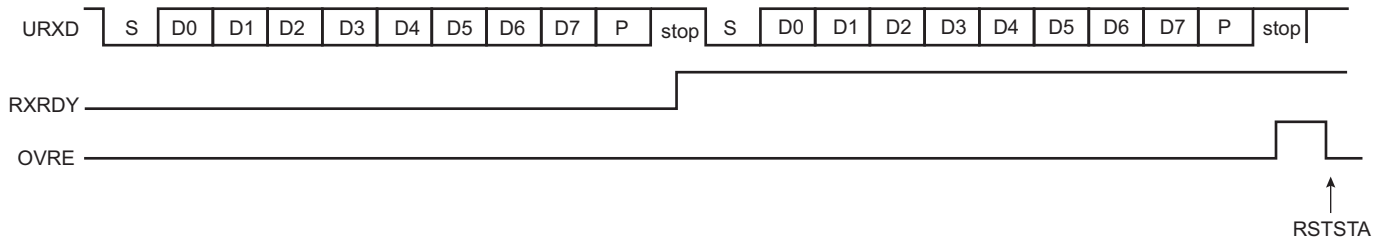


Downloaded from Arrow.com.

46.5.2.4 Receiver Overrun

The OVRE status bit in UART_SR is set if UART_RHR has not been read by the software (or the DMA Controller) since the last transfer, the RXRDY bit is still set and a new character is received. OVRE is cleared when the software writes a 1 to the bit RSTSTA (Reset Status) in UART_CR.

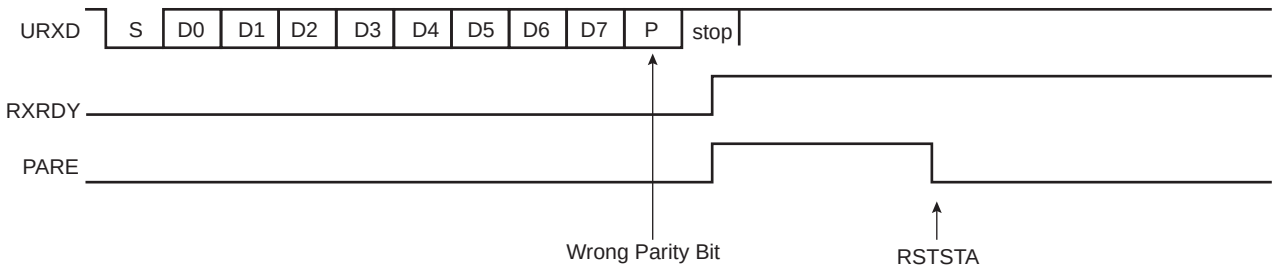
Figure 46-7. Receiver Overrun



46.5.2.5 Parity Error

Each time a character is received, the receiver calculates the parity of the received data bits, in accordance with the field PAR in the Mode Register (UART_MR). It then compares the result with the received parity bit. If different, the parity error bit PARE in UART_SR is set at the same time RXRDY is set. The parity bit is cleared when UART_CR is written with the bit RSTSTA (Reset Status) at 1. If a new character is received before the reset status command is written, the PARE bit remains at 1.

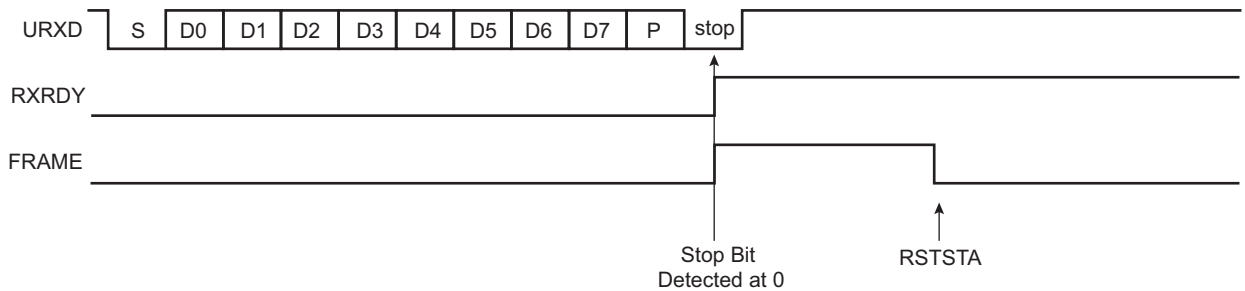
Figure 46-8. Parity Error



46.5.2.6 Receiver Framing Error

When a start bit is detected, it generates a character reception when all the data bits have been sampled. The stop bit is also sampled and when it is detected at 0, the FRAME (Framing Error) bit in UART_SR is set at the same time the RXRDY bit is set. The FRAME bit remains high until the Control Register (UART_CR) is written with the bit RSTSTA at 1.

Figure 46-9. Receiver Framing Error



Downloaded from Arrow.com.

46.5.2.7 Receiver Digital Filter

The UART embeds a digital filter on the receive line. It is disabled by default and can be enabled by writing a logical 1 in the FILTER bit of UART_MR. When enabled, the receive line is sampled using the 16x bit clock and a three-sample filter (majority 2 over 3) determines the value of the line.

46.5.3 Transmitter

46.5.3.1 Transmitter Reset, Enable and Disable

After device reset, the UART transmitter is disabled and must be enabled before being used. The transmitter is enabled by writing UART_CR with the bit TXEN at 1. From this command, the transmitter waits for a character to be written in the Transmit Holding Register (UART_THR) before actually starting the transmission.

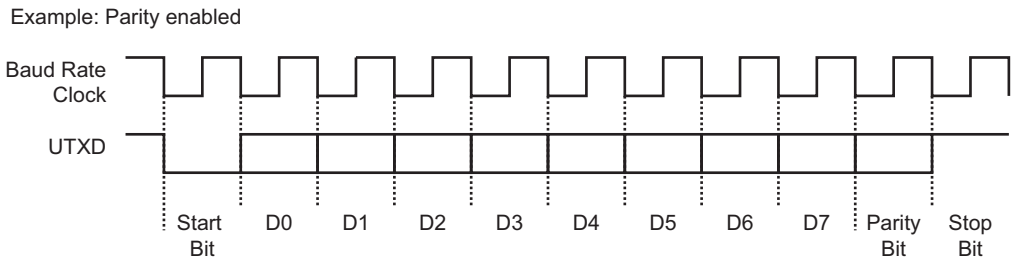
The programmer can disable the transmitter by writing UART_CR with the bit TXDIS at 1. If the transmitter is not operating, it is immediately stopped. However, if a character is being processed into the internal shift register and/or a character has been written in the UART_THR, the characters are completed before the transmitter is actually stopped.

The programmer can also put the transmitter in its reset state by writing the UART_CR with the bit RSTTX at 1. This immediately stops the transmitter, whether or not it is processing characters.

46.5.3.2 Transmit Format

The UART transmitter drives the pin UTXD at the baud rate clock speed. The line is driven depending on the format defined in UART_MR and the data stored in the internal shift register. One start bit at level 0, then the 8 data bits, from the lowest to the highest bit, one optional parity bit and one stop bit at 1 are consecutively shifted out as shown in the following figure. The field PARE in UART_MR defines whether or not a parity bit is shifted out. When a parity bit is enabled, it can be selected between an odd parity, an even parity, or a fixed space or mark bit.

Figure 46-10. Character Transmission



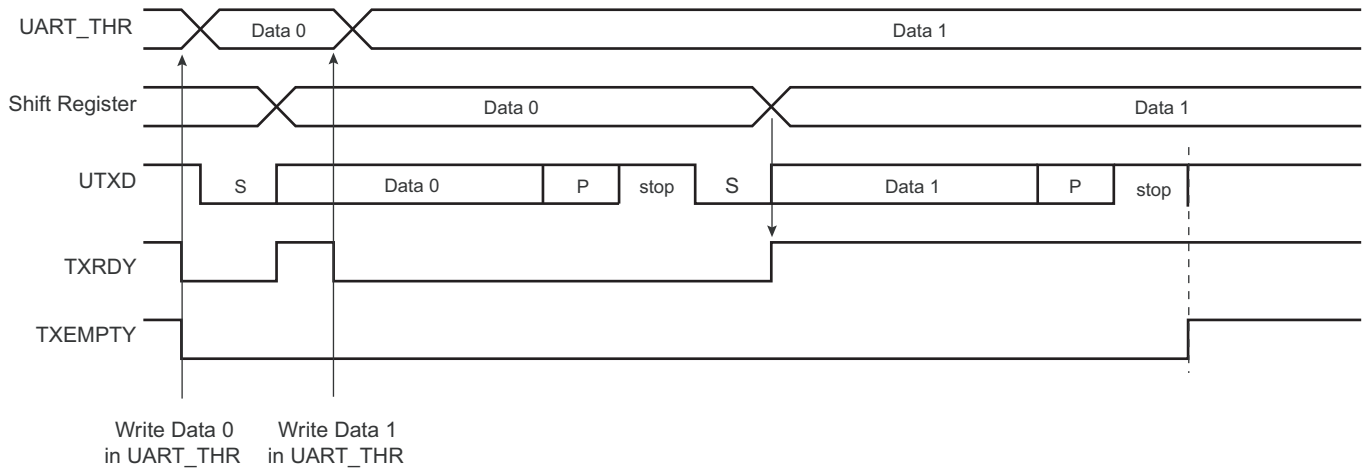
46.5.3.3 Transmitter Control

When the transmitter is enabled, the bit TXRDY (Transmitter Ready) is set in UART_SR. The transmission starts when the programmer writes in the UART_THR, and after the written character is transferred from UART_THR to the internal shift register. The TXRDY bit remains high until a second character is written in UART_THR. As soon as the first character is completed, the last character written in UART_THR is transferred into the internal shift register and TXRDY rises again, showing that the holding register is empty.

When both the internal shift register and UART_THR are empty, i.e., all the characters written in UART_THR have been processed, the TXEMPTY bit rises after the last stop bit has been completed.

Downloaded from Arrow.com.

Figure 46-11. Transmitter Control



46.5.4 DMA Support

Both the receiver and the transmitter of the UART are connected to a DMA Controller (DMAC) channel.

The DMA Controller channels are programmed via registers that are mapped within the DMAC user interface.

46.5.5 Comparison Function on Received Character

When a comparison is performed on a received character, the result of the comparison is reported on the CMP flag in UART_SR when UART_RHR is loaded with the new received character. The CMP flag is cleared by writing a one to the RSTSTA bit in UART_CR.

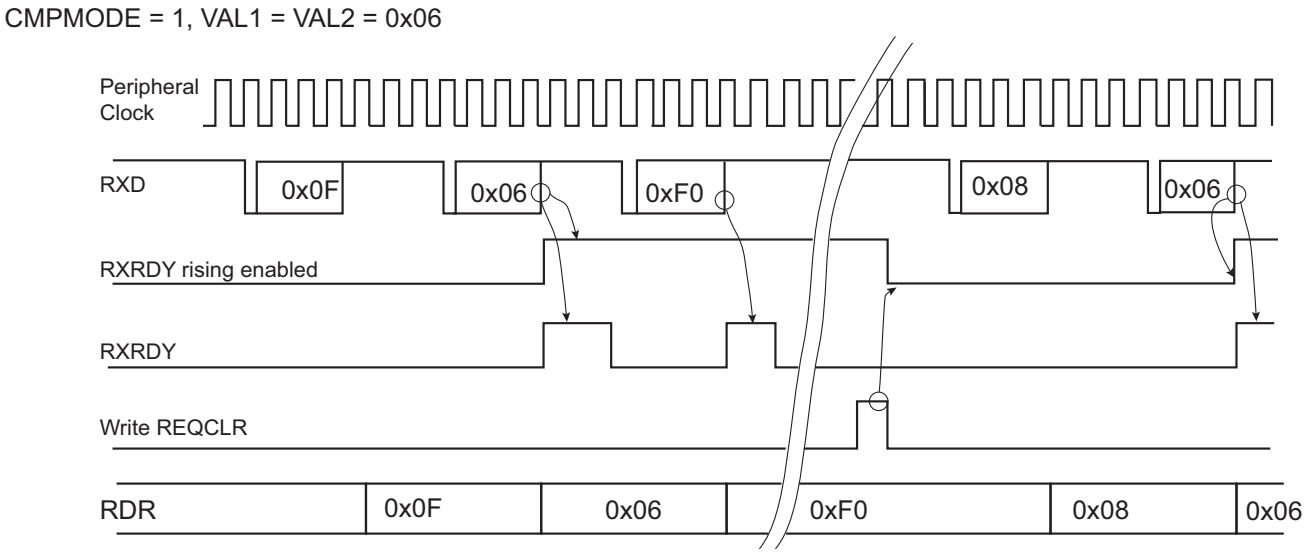
UART_CMPR (see Section 46.6.10 on page 1282) can be programmed to provide different comparison methods. These are listed below:

- If VAL1 equals VAL2, then the comparison is performed on a single value and the flag is set to 1 if the received character equals VAL1.
- If VAL1 is strictly lower than VAL2, then any value between VAL1 and VAL2 sets the CMP flag.
- If VAL1 is strictly higher than VAL2, then the flag CMP is set to 1 if either received character equals VAL1 or VAL2.

By programming the CMPMODE bit to 1, the comparison function result triggers the start of the loading of UART_RHR (see Figure 46-12). The trigger condition occurs as soon as the received character value matches the condition defined by the programming of VAL1, VAL2 and CMPPAR in UART_CMPR. The comparison trigger event can be restarted by writing a one to the REQCLR bit in UART_CR.

Downloaded from Arrow.com.

Figure 46-12. Receive Holding Register Management



46.5.6 Asynchronous and Partial Wake-up (SleepWalking)

Asynchronous and partial wake-up (SleepWalking) is a means of data pre-processing that qualifies an incoming event, thus allowing the UART to decide whether or not to wake up the system. SleepWalking is used primarily when the system is in Wait mode (refer to Section 31. "Power Management Controller (PMC)") but can also be enabled when the system is fully running.

No access must be performed in the UART between the enable of asynchronous partial wake-up and the wake-up performed by the UART.

If the system is in Wait mode and asynchronous and partial wake-up is enabled, the maximum baud rate that can be achieved equals 19200.

If the system is running or in Sleep mode, the maximum baud rate that can be achieved equals 115200 or higher. This limit is bounded by the peripheral clock frequency divided by 16.

The UART_RHR must be read before enabling asynchronous and partial wake-up.

When SleepWalking is enabled for the UART (see Section 31. "Power Management Controller (PMC)"), the PMC decodes a clock request from the UART. The request is generated as soon as there is a falling edge on the RXD line as this may indicate the beginning of a start bit. If the system is in Wait mode (processor and peripheral clocks switched off), the PMC restarts the fast RC oscillator and provides the clock only to the UART.

As soon as the clock is provided by the PMC, the UART processes the received frame and compares the received character with VAL1 and VAL2 in UART_CMPR (Section 46.6.10 on page 1282).

The UART instructs the PMC to disable the clock if the received character value does not meet the conditions defined by VAL1 and VAL2 fields in UART_CMPR (see Figure 46-14 on page 1269).

If the received character value meets the conditions, the UART instructs the PMC to exit the full system from Wait mode (see Figure 46-13 on page 1268).

The VAL1 and VAL2 fields can be programmed to provide different comparison methods and thus matching conditions.

- If VAL1 equals VAL2, then the comparison is performed on a single value and the wake-up is triggered if the received character equals VAL1.
- If VAL1 is strictly lower than VAL2, then any value between VAL1 and VAL2 wakes up the system.
- If VAL1 is strictly higher than VAL2, then the wake-up is triggered if the received character equals VAL1 or VAL2.

Downloaded from Arrow.com.

- If VAL1 = 0 and VAL2 = 255, the wake-up is triggered as soon as a character is received.

The matching condition can be configured to include the parity bit (CMPPAR in UART_CMPR). Thus, if the received data matches the comparison condition defined by VAL1 and VAL2 but a parity error is encountered, the matching condition is cancelled and the UART instructs the PMC to disable the clock (see Figure 46-14 on page 1269).

If the processor and peripherals are running, the UART can be configured in Asynchronous and partial wake-up mode by enabling the PMC_SLPWK_ER (see Section 31. ”Power Management Controller (PMC)”). When activity is detected on the receive line, the UART requests the clock from the PMC and the comparison is performed. If there is a comparison match, the UART continues to request the clock. If there is no match, the clock is switched off for the UART only, until a new activity is detected.

The CMPMODE configuration has no effect when Asynchronous and partial wake-up mode is enabled for the UART (see PMC_SLPWK_ER in Section 31. ”Power Management Controller (PMC)”).

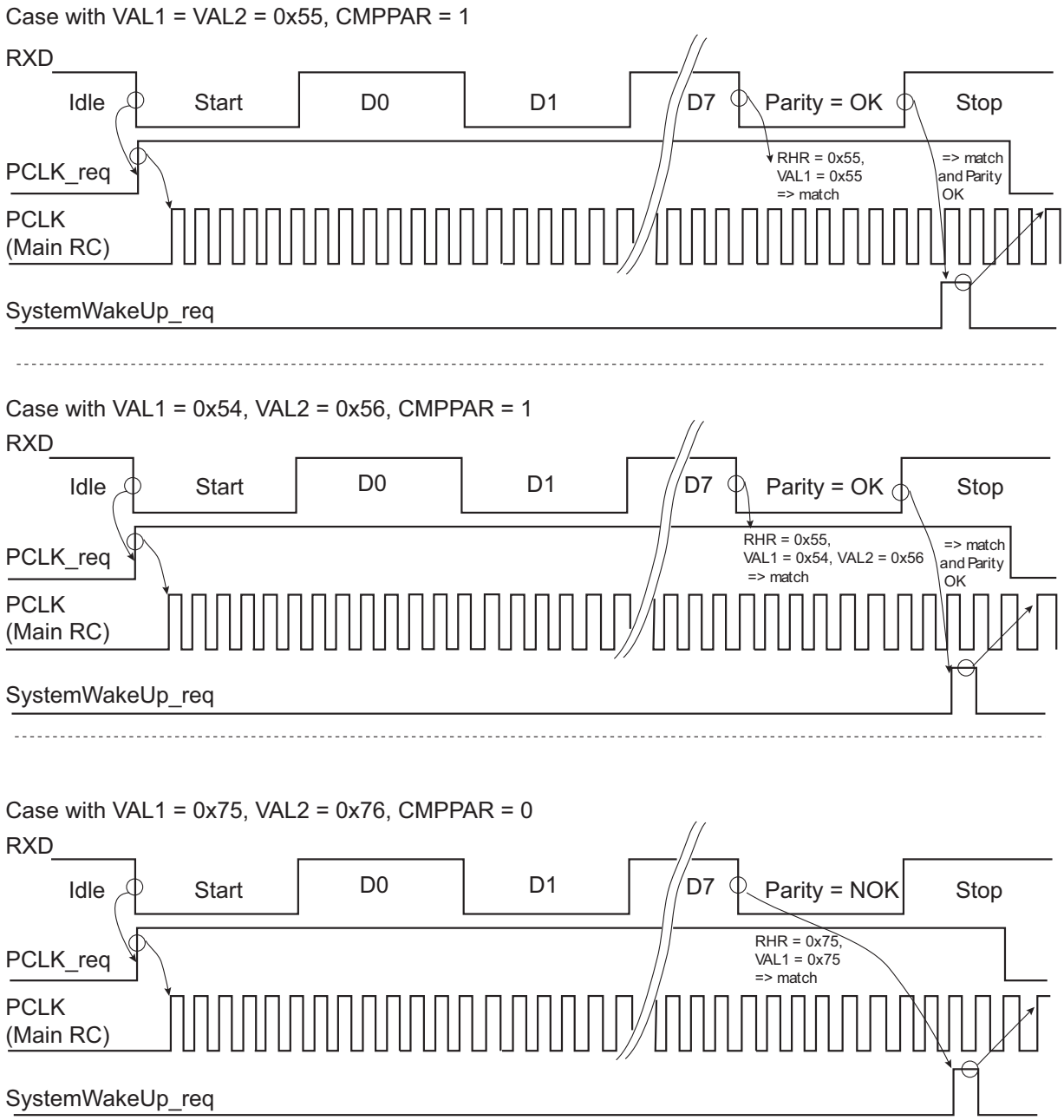
When the system is kept in active/running mode and the UART enters Asynchronous and partial wake-up mode, the flag CMP must be programmed as the unique source of the UART interrupt.

When the system exits Wait mode as the result of a matching condition, the RXRDY flag is used to determine if the UART is the source of exit.

Note: If the SleepWalking function is enabled on the UART, a divide by 8 of the peripheral clock versus the bus clock is not possible. Other dividers can be used with no constraints.

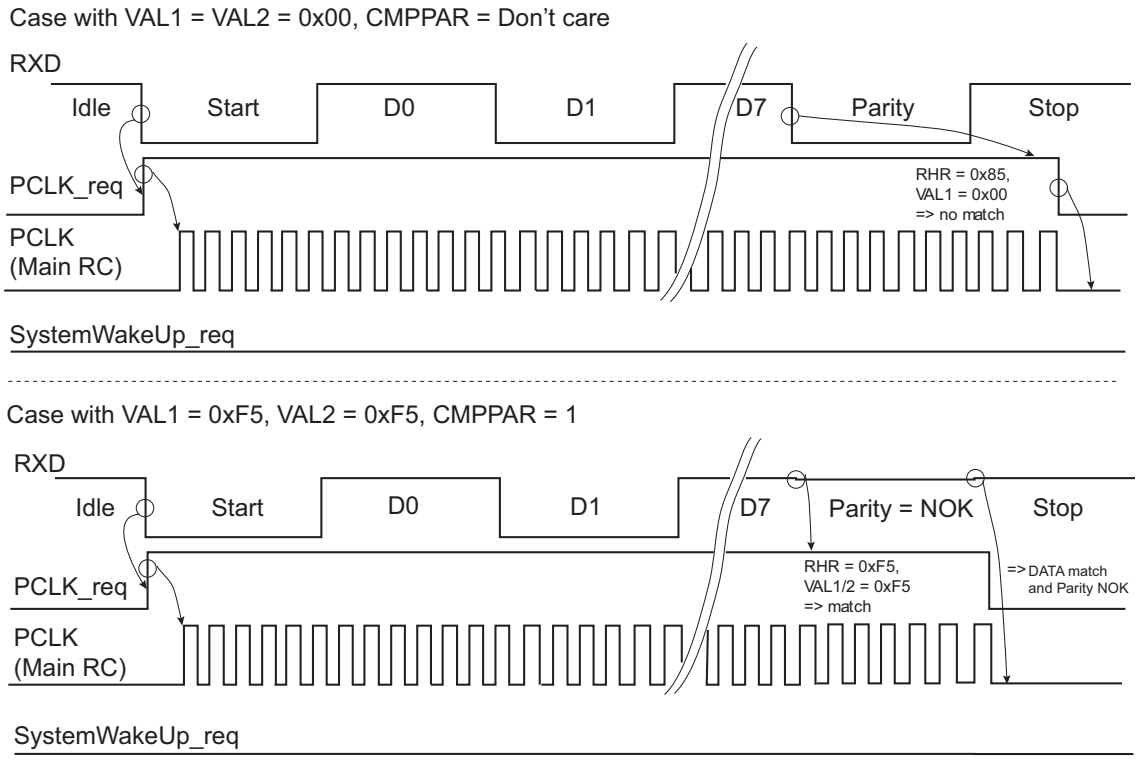
Downloaded from Arrow.com.

Figure 46-13. Asynchronous Wake-up Use Case Examples



Downloaded from Arrow.com.

Figure 46-14. Asynchronous Event Generating Only Partial Wake-up



46.5.7 Register Write Protection

To prevent any single software error from corrupting UART behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the UART Write Protection Mode Register (UART_WPMR).

The following registers can be write-protected:

- UART Mode Register
- UART Baud Rate Generator Register
- UART Comparison Register

46.5.8 Test Modes

The UART supports three test modes. These modes of operation are programmed by using the CHMODE field in UART_MR.

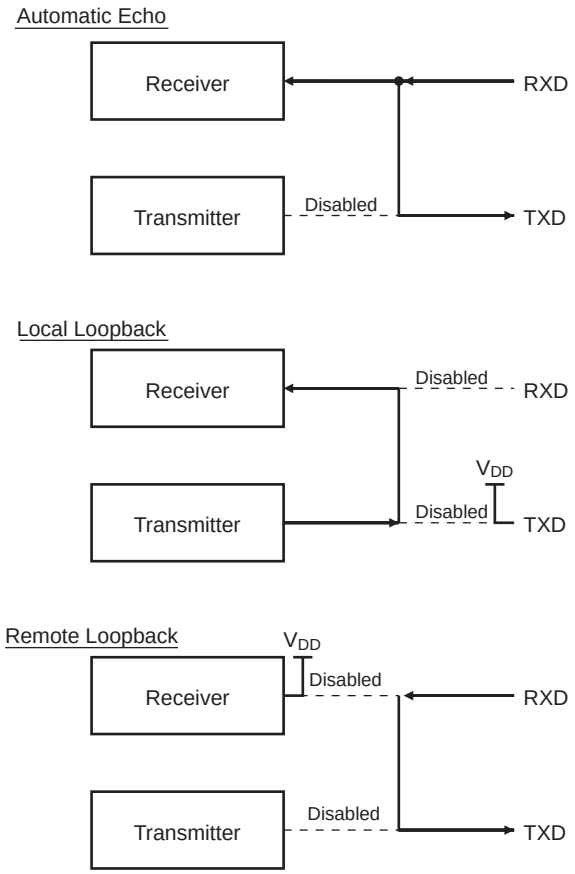
The Automatic echo mode allows a bit-by-bit retransmission. When a bit is received on the URXD line, it is sent to the UTXD line. The transmitter operates normally, but has no effect on the UTXD line.

The Local loopback mode allows the transmitted characters to be received. UTXD and URXD pins are not used and the output of the transmitter is internally connected to the input of the receiver. The URXD pin level has no effect and the UTXD line is held high, as in idle state.

The Remote loopback mode directly connects the URXD pin to the UTXD line. The transmitter and the receiver are disabled and have no effect. This mode allows a bit-by-bit retransmission.

Downloaded from Arrow.com.

Figure 46-15. Test Modes



Downloaded from Arrow.com.

46.6 Universal Asynchronous Receiver Transmitter (UART) User Interface

Table 46-4. Register Mapping

Offset	Register	Name	Access	Reset
0x0000	Control Register	UART_CR	Write-only	–
0x0004	Mode Register	UART_MR	Read/Write	0x0
0x0008	Interrupt Enable Register	UART_IER	Write-only	–
0x000C	Interrupt Disable Register	UART_IDR	Write-only	–
0x0010	Interrupt Mask Register	UART_IMR	Read-only	0x0
0x0014	Status Register	UART_SR	Read-only	–
0x0018	Receive Holding Register	UART_RHR	Read-only	0x0
0x001C	Transmit Holding Register	UART_THR	Write-only	–
0x0020	Baud Rate Generator Register	UART_BRGR	Read/Write	0x0
0x0024	Comparison Register	UART_CMPR	Read/Write	0x0
0x0028–0x003C	Reserved	–	–	–
0x0040–0x00E0	Reserved	–	–	–
0x00E4	Write Protection Mode Register	UART_WPMR	Read/Write	0x0
0x00E8	Reserved	–	–	–
0x00EC–0x00FC	Reserved	–	–	–

Downloaded from Arrow.com.

46.6.1 UART Control Register

Name: UART_CR

Address: 0x400E0800 (0), 0x400E0A00 (1), 0x400E1A00 (2), 0x400E1C00 (3), 0x400E1E00 (4)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	REQCLR	–	–	–	RSTSTA
7	6	5	4	3	2	1	0
TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	–	–

- **RSTRX: Reset Receiver**

0: No effect.

1: The receiver logic is reset and disabled. If a character is being received, the reception is aborted.

- **RSTTX: Reset Transmitter**

0: No effect.

1: The transmitter logic is reset and disabled. If a character is being transmitted, the transmission is aborted.

- **RXEN: Receiver Enable**

0: No effect.

1: The receiver is enabled if RXDIS is 0.

- **RXDIS: Receiver Disable**

0: No effect.

1: The receiver is disabled. If a character is being processed and RSTRX is not set, the character is completed before the receiver is stopped.

- **TXEN: Transmitter Enable**

0: No effect.

1: The transmitter is enabled if TXDIS is 0.

- **TXDIS: Transmitter Disable**

0: No effect.

1: The transmitter is disabled. If a character is being processed and a character has been written in the UART_THR and RSTTX is not set, both characters are completed before the transmitter is stopped.

- **RSTSTA: Reset Status**

0: No effect.

1: Resets the status bits PARE, FRAME, CMP and OVRE in the UART_SR.

Downloaded from Arrow.com.

- **REQCLR: Request Clear**

SleepWalking enabled:

0: No effect.

1: Bit REQCLR clears the potential clock request currently issued by UART, thus the potential system wake-up is cancelled.

SleepWalking disabled:

0: No effect.

1: Bit REQCLR restarts the comparison trigger to enable receive holding register loading.

Downloaded from Arrow.com.

46.6.2 UART Mode Register

Name: UART_MR

Address: 0x400E0804 (0), 0x400E0A04 (1), 0x400E1A04 (2), 0x400E1C04 (3), 0x400E1E04 (4)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CHMODE		–	BRSRCCK	PAR			–
7	6	5	4	3	2	1	0
–	–	–	FILTER	–	–	–	–

- **FILTER: Receiver Digital Filter**

0 (DISABLED): UART does not filter the receive line.

1 (ENABLED): UART filters the receive line using a three-sample filter (16x-bit clock) (2 over 3 majority).

- **PAR: Parity Type**

Value	Name	Description
0	EVEN	Even Parity
1	ODD	Odd Parity
2	SPACE	Space: parity forced to 0
3	MARK	Mark: parity forced to 1
4	NO	No parity

- **BRSRCCK: Baud Rate Source Clock**

0 (PERIPH_CLK): The baud rate is driven by the peripheral clock

1 (PMC_PCK): The baud rate is driven by a PMC programmable clock PCK (see Section 31. ”Power Management Controller (PMC)”).

- **CHMODE: Channel Mode**

Value	Name	Description
0	NORMAL	Normal mode
1	AUTOMATIC	Automatic echo
2	LOCAL_LOOPBACK	Local loopback
3	REMOTE_LOOPBACK	Remote loopback

Downloaded from Arrow.com.

46.6.3 UART Interrupt Enable Register

Name: UART_IER

Address: 0x400E0808 (0), 0x400E0A08 (1), 0x400E1A08 (2), 0x400E1C08 (3), 0x400E1E08 (4)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CMP	–	–	–	–	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

- **RXRDY: Enable RXRDY Interrupt**
- **TXRDY: Enable TXRDY Interrupt**
- **OVRE: Enable Overrun Error Interrupt**
- **FRAME: Enable Framing Error Interrupt**
- **PARE: Enable Parity Error Interrupt**
- **TXEMPTY: Enable TXEMPTY Interrupt**
- **CMP: Enable Comparison Interrupt**

Downloaded from Arrow.com.

46.6.4 UART Interrupt Disable Register

Name: UART_IDR

Address: 0x400E080C (0), 0x400E0A0C (1), 0x400E1A0C (2), 0x400E1C0C (3), 0x400E1E0C (4)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CMP	–	–	–	–	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

- **RXRDY: Disable RXRDY Interrupt**
- **TXRDY: Disable TXRDY Interrupt**
- **OVRE: Disable Overrun Error Interrupt**
- **FRAME: Disable Framing Error Interrupt**
- **PARE: Disable Parity Error Interrupt**
- **TXEMPTY: Disable TXEMPTY Interrupt**
- **CMP: Disable Comparison Interrupt**

Downloaded from Arrow.com.

46.6.5 UART Interrupt Mask Register

Name: UART_IMR

Address: 0x400E0810 (0), 0x400E0A10 (1), 0x400E1A10 (2), 0x400E1C10 (3), 0x400E1E10 (4)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CMP	–	–	–	–	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

- **RXRDY: Mask RXRDY Interrupt**
- **TXRDY: Disable TXRDY Interrupt**
- **OVRE: Mask Overrun Error Interrupt**
- **FRAME: Mask Framing Error Interrupt**
- **PARE: Mask Parity Error Interrupt**
- **TXEMPTY: Mask TXEMPTY Interrupt**
- **CMP: Mask Comparison Interrupt**

Downloaded from Arrow.com.

46.6.6 UART Status Register

Name: UART_SR

Address: 0x400E0814 (0), 0x400E0A14 (1), 0x400E1A14 (2), 0x400E1C14 (3), 0x400E1E14 (4)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CMP	–	–	–	–	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	–	TXRDY	RXRDY

- **RXRDY: Receiver Ready**

0: No character has been received since the last read of the UART_RHR, or the receiver is disabled.

1: At least one complete character has been received, transferred to UART_RHR and not yet read.

- **TXRDY: Transmitter Ready**

0: A character has been written to UART_THR and not yet transferred to the internal shift register, or the transmitter is disabled.

1: There is no character written to UART_THR not yet transferred to the internal shift register.

- **OVRE: Overrun Error**

0: No overrun error has occurred since the last RSTSTA.

1: At least one overrun error has occurred since the last RSTSTA.

- **FRAME: Framing Error**

0: No framing error has occurred since the last RSTSTA.

1: At least one framing error has occurred since the last RSTSTA.

- **PARE: Parity Error**

0: No parity error has occurred since the last RSTSTA.

1: At least one parity error has occurred since the last RSTSTA.

- **TXEMPTY: Transmitter Empty**

0: There are characters in UART_THR, or characters being processed by the transmitter, or the transmitter is disabled.

1: There are no characters in UART_THR and there are no characters being processed by the transmitter.

- **CMP: Comparison Match**

0: No received character matches the comparison criteria programmed in VAL1, VAL2 fields and in CMPPAR bit since the last RSTSTA.

1: The received character matches the comparison criteria.

Downloaded from Arrow.com.

46.6.7 UART Receiver Holding Register

Name: UART_RHR

Address: 0x400E0818 (0), 0x400E0A18 (1), 0x400E1A18 (2), 0x400E1C18 (3), 0x400E1E18 (4)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
RXCHR							

- **RXCHR: Received Character**

Last received character if RXRDY is set.

Downloaded from Arrow.com.

46.6.8 UART Transmit Holding Register

Name: UART_THR

Address: 0x400E081C (0), 0x400E0A1C (1), 0x400E1A1C (2), 0x400E1C1C (3), 0x400E1E1C (4)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
TXCHR							

- **TXCHR: Character to be Transmitted**

Next character to be transmitted after the current character if TXRDY is not set.

Downloaded from Arrow.com.

46.6.9 UART Baud Rate Generator Register

Name: UART_BRGR

Address: 0x400E0820 (0), 0x400E0A20 (1), 0x400E1A20 (2), 0x400E1C20 (3), 0x400E1E20 (4)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CD							
7	6	5	4	3	2	1	0
CD							

- **CD: Clock Divisor**

0: Baud rate clock is disabled

1 to 65,535:

If BRSRCCK = 0:

$$CD = \frac{f_{peripheral\ clock}}{16 \times Baud\ Rate}$$

If BRSRCCK = 1:

$$CD = \frac{f_{PCKx}}{16 \times Baud\ Rate}$$

Downloaded from Arrow.com.

46.6.10 UART Comparison Register

Name: UART_CMPR

Address: 0x400E0824 (0), 0x400E0A24 (1), 0x400E1A24 (2), 0x400E1C24 (3), 0x400E1E24 (4)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
VAL2							
15	14	13	12	11	10	9	8
–	CMPPAR	–	CMPMODE	–	–	–	–
7	6	5	4	3	2	1	0
VAL1							

- **VAL1: First Comparison Value for Received Character**

0–255: The received character must be higher or equal to the value of VAL1 and lower or equal to VAL2 to set CMP flag in UART_SR. If asynchronous partial wake-up (SleepWalking) is enabled in PMC_SLPWK_ER, the UART requests a system wake-up if the condition is met.

- **CMPMODE: Comparison Mode**

Value	Name	Description
0	FLAG_ONLY	Any character is received and comparison function drives CMP flag.
1	START_CONDITION	Comparison condition must be met to start reception.

- **CMPPAR: Compare Parity**

0: The parity is not checked and a bad parity cannot prevent from waking up the system.

1: The parity is checked and a matching condition on data can be cancelled by an error on parity bit, so no wake-up is performed.

- **VAL2: Second Comparison Value for Received Character**

0–255: The received character must be lower or equal to the value of VAL2 and higher or equal to VAL1 to set CMP flag in UART_SR. If asynchronous partial wake-up (SleepWalking) is enabled in PMC_SLPWK_ER, the UART requests a system wake-up if condition is met.

Downloaded from Arrow.com.

46.6.11 UART Write Protection Mode Register

Name: UART_WPMR

Address: 0x400E08E4 (0), 0x400E0AE4 (1), 0x400E1AE4 (2), 0x400E1CE4 (3), 0x400E1EE4 (4)

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							
23	22	21	20	19	18	17	16
WPKEY							
15	14	13	12	11	10	9	8
WPKEY							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x554152 (UART in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x554152 (UART in ASCII).

See Section 46.5.7 ”Register Write Protection” for the list of registers that can be protected.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x554152	PASSWD	Writing any other value in this field aborts the write operation. Always reads as 0.

Downloaded from Arrow.com.

47. Controller Area Network (MCAN)

47.1 Description

The Controller Area Network (MCAN) performs communication according to ISO11898-1 (Bosch CAN specification 2.0 part A,B) and to Bosch CAN FD specification V1.0. Additional transceiver hardware is required for connection to the physical layer.

All functions concerning the handling of messages are implemented by the Rx Handler and the Tx Handler. The Rx Handler manages message acceptance filtering, the transfer of received messages from the CAN core to the Message RAM, as well as providing receive message status information. The Tx Handler is responsible for the transfer of transmit messages from the Message RAM to the CAN core, as well as providing transmit status information.

Acceptance filtering is implemented by a combination of up to 128 filter elements, where each element can be configured as a range, as a bit mask, or as a dedicated ID filter.

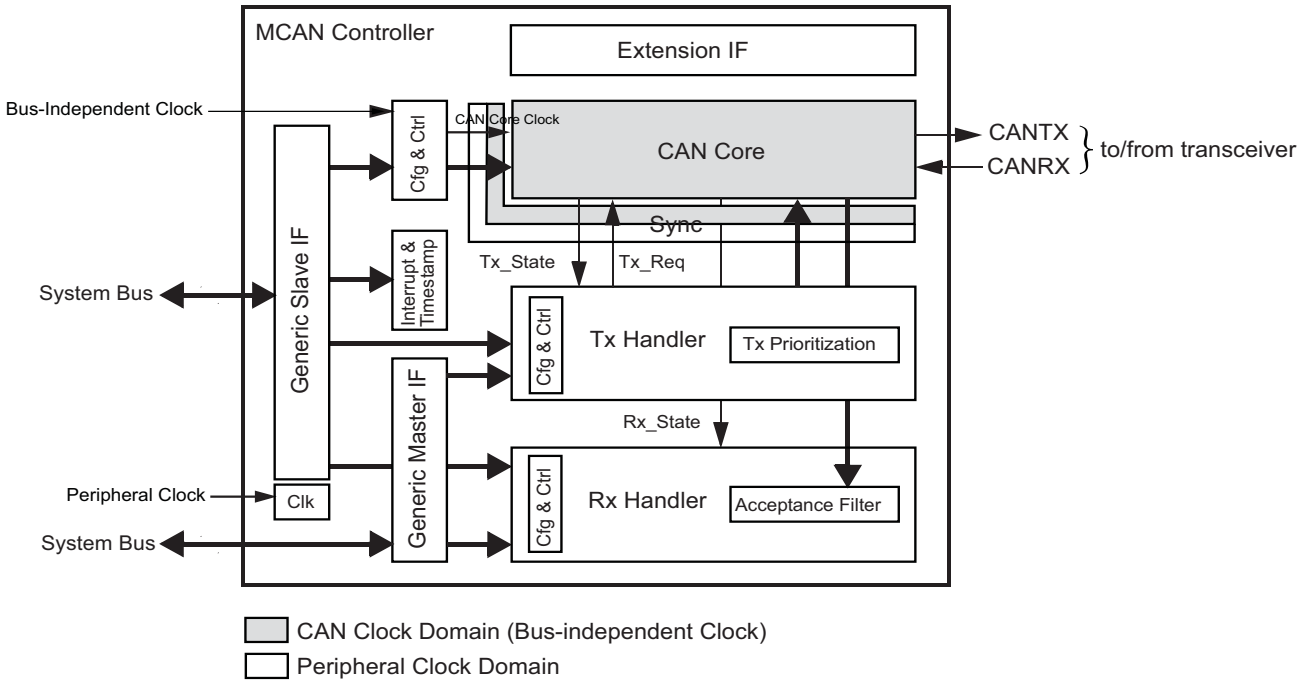
47.2 Embedded Characteristics

- Compliant with CAN Protocol Version 2.0 Part A, B and ISO 11898-1
- CAN FD with up to 64 Data Bytes Supported
- CAN Error Logging
- AUTOSAR Optimized
- SAE J1939 Optimized
- Improved Acceptance Filtering
- Two Configurable Receive FIFOs
- Separate Signalling on Reception of High Priority Messages
- Up to 64 Dedicated Receive Buffers
- Up to 32 Dedicated Transmit Buffers
- Configurable Transmit FIFO
- Configurable Transmit Queue
- Configurable Transmit Event FIFO
- Direct Message RAM Access for Processor
- Multiple MCANs May Share the Same Message RAM
- Programmable Loop-back Test Mode
- Maskable Module Interrupts
- Support for Asynchronous CAN and System Bus Clocks
- Power-down Support
- Debug on CAN Support

Downloaded from Arrow.com.

47.3 Block Diagram

Figure 47-1. MCAN Block Diagram



Note: Refer to Section 31. "Power Management Controller (PMC)" for details about the bus-independent clock (PCK5).

Downloaded from Arrow.com.

47.4 Product Dependencies

47.4.1 I/O Lines

The pins used to interface to the compliant external devices can be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the CAN pins to their peripheral functions.

Table 47-1. I/O Lines

Instance	Signal	I/O Line	Peripheral
MCAN0	CANRX0	PB3	A
MCAN0	CANTX0	PB2	A
MCAN1	CANRX1	PC12	C
MCAN1	CANRX1	PD28	B
MCAN1	CANTX1	PC14	C
MCAN1	CANTX1	PD12	B

47.4.2 Power Management

The MCAN can be clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the MCAN clock.

In order to achieve a stable function of the MCAN, the system bus clock must always be faster than or equal to the CAN clock.

It is recommended to use the CAN clock at frequencies of 20, 40 or 80 MHz. To achieve these frequencies, PMC PCK5 must select the UPLLCK (480 MHz) as source clock and divide by 24,12, or 6. PCK5 allows the system bus and processor clock to be modified without affecting the bit rate communication.

47.4.3 Interrupt Sources

The two MCAN interrupt lines (m_can_int0, m_can_int1) are connected on internal sources of the Interrupt Controller.

Using the MCAN interrupts requires the Interrupt Controller to be programmed first.

Interrupt sources can be routed either to m_can_int0 or to m_can_int1. By default all interrupt sources are routed to interrupt line m_can_int0. By programming MCAN_ILE.EINT0 and MCAN_ILE.EINT1, the interrupt sources can be enabled or disabled separately.

Table 47-2. Peripheral IDs

Instance	ID
MCAN0	35
MCAN1	37

47.4.4 Address Configuration

The LSBs [bits 15:2] for each section of the CAN Message RAM are configured in the respective buffer configuration registers.

The MSBs [bits 31:16] of the CAN Message RAM for CAN0 and CAN1 are configured in CCFG_CAN0 and CCFG_SYSIO registers.

Downloaded from Arrow.com.

47.5 Functional Description

47.5.1 Operating Modes

47.5.1.1 Software Initialization

Software initialization is started by setting bit MCAN_CCCR.INIT, either by software or by a hardware reset, when an uncorrected bit error was detected in the Message RAM, or by going Bus_Off. While MCAN_CCCR.INIT is set, message transfer from and to the CAN bus is stopped and the status of the CAN bus output CANTX is recessive (HIGH). The counters of the Error Management Logic EML are unchanged. Setting MCAN_CCCR.INIT does not change any configuration register. Resetting MCAN_CCCR.INIT finishes the software initialization. Afterwards the Bit Stream Processor BSP synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (≡ Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the MCAN configuration registers is only enabled when both bits MCAN_CCCR.INIT and MCAN_CCCR.CCE are set (protected write).

MCAN_CCCR.CCE can only be configured when MCAN_CCCR.INIT = '1'. MCAN_CCCR.CCE is automatically cleared when MCAN_CCCR.INIT = '0'.

The following registers are cleared when MCAN_CCCR.CCE = '1':

- High Priority Message Status (MCAN_HPMS)
- Receive FIFO 0 Status (MCAN_RXF0S)
- Receive FIFO 1 Status (MCAN_RXF1S)
- Transmit FIFO/Queue Status (MCAN_TXFQS)
- Transmit Buffer Request Pending (MCAN_TXBRP)
- Transmit Buffer Transmission Occurred (MCAN_TXBTO)
- Transmit Buffer Cancellation Finished (MCAN_TXBCF)
- Transmit Event FIFO Status (MCAN_TXEFS)

The Timeout Counter value MCAN_TOCV.TOC is loaded with the value configured by MCAN_TOCC.TOP when MCAN_CCCR.CCE = '1'.

In addition, the state machines of the Tx Handler and Rx Handler are held in idle state while MCAN_CCCR.CCE = '1'.

The following registers are only writeable while MCAN_CCCR.CCE = '0'

- Transmit Buffer Add Request (MCAN_TXBAR)
- Transmit Buffer Cancellation Request (MCAN_TXBCR)

MCAN_CCCR.TEST and MCAN_CCCR.MON can only be set when MCAN_CCCR.INIT = '1' and MCAN_CCCR.CCE = '1'. Both bits may be cleared at any time. MCAN_CCCR.DAR can only be configured when MCAN_CCCR.INIT = '1' and MCAN_CCCR.CCE = '1'.

47.5.1.2 Normal Operation

Once the MCAN is initialized and MCAN_CCCR.INIT is cleared, the MCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including Message ID and DLC are stored into a dedicated Rx Buffer or into Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted, dedicated Tx Buffers and/or a Tx FIFO or a Tx Queue can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

Downloaded from Arrow.com.

47.5.1.3 CAN FD Operation

There are two variants in the CAN FD frame format, first the CAN FD frame without bit rate switching where the data field of a CAN frame may be longer than 8 bytes. The second variant is the CAN FD frame where control field, data field, and CRC field of a CAN frame are transmitted with a higher bit rate than the beginning and the end of the frame.

The MCAN operation mode is enabled by configuring MCAN_CCCR.CME. In case MCAN_CCCR.CME = 1, transmission of long CAN FD frames and reception of long and fast CAN FD frames is enabled. With MCAN_CCCR.CME = 2 or 3, transmission and reception of long and fast CAN FD frames is enabled. MCAN_CCCR.CME can only be changed while MCAN_CCCR.INIT and MCAN_CCCR.CCE are both set.

When initialization is completed, the CAN FD protocol option is disabled and must be requested by writing to MCAN_CCCR.CMR.

A mode change requested by writing to MCAN_CCCR.CMR will be executed next time the CAN protocol controller FSM reaches idle phase between CAN frames. Upon this event MCAN_CCCR.CMR is cleared and the status flags MCAN_CCCR.FDBS and MCAN_CCCR.FDO are set accordingly. In case the requested CAN operation mode is not enabled, the value written to MCAN_CCCR.CMR is retained until it is overwritten by the next mode change request. Default is CAN operation according to ISO11898-1.

It is not necessary to change the CAN operation mode after system startup. A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case disable the CAN FD bit rate switching option for transmissions.
- During system startup all nodes are transmitting according to ISO11898-1 until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- End-of-line programming in case not all nodes are CAN FD-capable. Non-CAN FD nodes are held in silent mode until programming has completed. Then all nodes revert to CAN communication according ISO11898-1.

When MCAN_CCCR.CME ≠ 0, received CAN FD frames are interpreted according to the CAN FD Protocol Specification. The reserved bit in CAN frames with 11-bit identifiers and the first reserved bit in CAN frames with 29-bit identifiers will be decoded as EDL bit. EDL = recessive signifies a CAN FD frame, EDL = dominant signifies a standard CAN frame. In a CAN FD frame, the two bits following EDL, r0 and BRS, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by r0 = dominant and BRS = recessive. The coding of r0 = recessive is reserved for future expansion of the protocol.

Reception of CAN frames according to ISO 11898-1 is possible in all CAN operation modes.

Status bits MCAN_CCCR.FDO and MCAN_CCCR.FDBS indicate the format of transmitted frames. When MCAN_CCCR.FDO is set, frames will be transmitted in CAN FD format with EDL = recessive. When both MCAN_CCCR.FDO and MCAN_CCCR.FDBS are set, frames are transmitted in CAN FD format with bit rate switching and both bits EDL and BRS = recessive.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15, which in standard CAN all code a data field of 8 bytes, are coded according to Table 47-3 below.

Table 47-3. Coding of DLC in CAN FD

DLC	9	10	11	12	13	14	15
Number of Data Bytes	12	16	20	24	32	48	64

In CAN FD frames, the bit timing will be switched inside the frame, after the BRS (Bit Rate Switch) bit, if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the standard CAN bit timing is used as defined by the Bit Timing and Prescaler register (MCAN_BTP). In the following CAN FD data phase, the fast CAN bit timing is

Downloaded from Arrow.com.

used as defined by the Fast Bit Timing and Prescaler register (MCAN_FBTP). The bit timing reverts back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the CAN core clock frequency. Example: with a CAN clock frequency of 20 MHz and the shortest configurable bit time of 4 t_q, the bit rate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD long and CAN FD fast, the value of the bit ESI (Error Status Indicator) is determined by the transmitter's error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant.

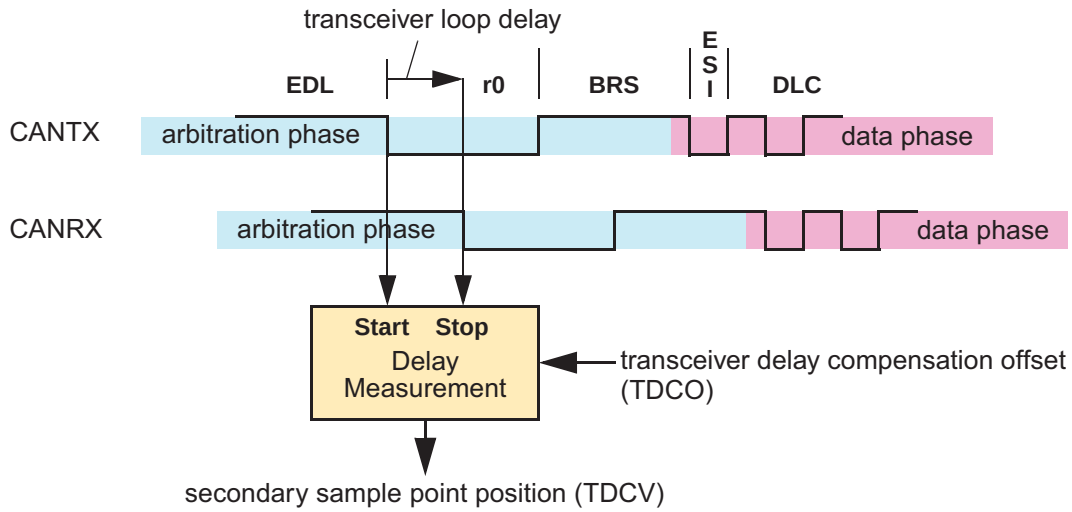
47.5.1.4 Transceiver Delay Compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin CANTX the protocol controller receives the transmitted data from its local CAN transceiver via pin CANRX. The received data is delayed by the CAN transceiver's loop delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. In order to enable a data phase bit time that is even shorter than the transceiver loop delay, the delay compensation is introduced. Without transceiver delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transceivers loop delay.

Description

The CAN FD protocol unit has implemented a delay compensation mechanism to compensate the CAN transceiver's loop delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver. Figure 47-2 below describes how the transceiver loop delay is measured.

Figure 47-2. Transceiver Delay Measurement



Within each CAN FD frame, the transmitter measures the delay between the data transmitted at pin CANTX and the data received at pin CANRX. The measurement is done once, at the falling edge of bit EDL to bit r0. The delay is measured in CAN core clock periods.

A secondary sample point position is calculated by adding a configurable transceiver delay compensation offset MCAN_FBTP.TDCO to the measured transceiver delay. This transceiver delay compensation value MCAN_TEST.TDCV is the sum of the measured transceiver delay and the transceiver delay compensation offset. The transceiver delay compensation offset is chosen to adjust the secondary sample point inside the bit time (e.g. half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of time quanta t_q.

Downloaded from Arrow.com.

To check for bit errors during the data phase, the delayed transmit data is compared against the received data at the secondary sample point. If a bit error is detected at the secondary sample point, the transmitter will react to this bit error at the next following regular sample point. During arbitration phase the delay compensation is always disabled.

For the transceiver delay compensation the following boundary conditions have to be considered:

- The sum of the measured delay from CANTX to CANRX and the configured transceiver delay compensation offset MCAN_FBTP.TDCO has to be less than 3 bit times in the data phase.
- The sum of the measured delay from CANTX to CANRX and the configured transceiver delay compensation offset MCAN_FBTP.TDCO has to be less or equal 63 CAN core clock periods. In case this sum exceeds 63 CAN core clock periods, the maximum value of 63 CAN core clock periods is used for transceiver delay compensation.

The actual delay compensation value is monitored by reading MCAN_TEST.TDCV.

Configuration and Status

Compensation for the transceiver loop delay by the MCAN is enabled via MCAN_FBTP.TDC. The transceiver delay compensation offset is configured via MCAN_FBTP.TDCO. The actual delay compensation value applied by the MCAN's protocol engine can be read from MCAN_TEST.TDCV.

47.5.1.5 Restricted Operation Mode

In Restricted Operation mode, the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. The processor can set the MCAN into Restricted Operation mode by setting bit MCAN_CCCR.ASM. The bit can only be set by the processor when both MCAN_CCCR.CCE and MCAN_CCCR.INIT are set to '1'. The bit can be reset by the processor at any time.

Restricted Operation mode is automatically entered when the Tx Handler was not able to read data from the Message RAM in time. To leave Restricted Operation mode, the processor has to reset MCAN_CCCR.ASM.

The Restricted Operation mode can be used in applications that adapt themselves to different CAN bit rates. In this case the application tests different bit rates and leaves the Restricted Operation mode after it has received a valid frame.

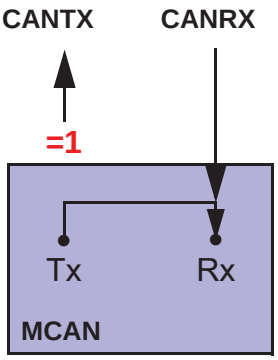
47.5.1.6 Bus Monitoring Mode

The MCAN is set in Bus Monitoring mode by setting MCAN_CCCR.MON. In Bus Monitoring mode (see ISO11898-1, 10.12 Bus monitoring), the MCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the MCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the MCAN monitors this dominant bit, although the CAN bus may remain in recessive state. In Bus Monitoring mode, the Tx Buffer Request Pending register (MCAN_TXBRP) is held in reset state.

The Bus Monitoring mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. Figure 47-4 shows the connection of signals CANTX and CANRX to the MCAN in Bus Monitoring mode.

Downloaded from Arrow.com.

Figure 47-3. Pin Control in Bus Monitoring Mode



Bus Monitoring Mode

47.5.1.7 Disabled Automatic Retransmission

According to the CAN Specification (see ISO11898-1, 6.3.3 Recovery Management), the MCAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled. To support time-triggered communication as described in ISO 11898-1, chapter 9.2, the automatic retransmission may be disabled via MCAN_CCCR.DAR.

Frame Transmission in DAR Mode

In DAR mode all transmissions are automatically cancelled after they started on the CAN bus. A Tx Buffer's Tx Request Pending bit TXBRP.TRPx is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  Corresponding Tx Buffer Transmission Occurred bit MCAN_TXBTO.TOx set
  Corresponding Tx Buffer Cancellation Finished bit MCAN_TXBCF.CFx not set
- Successful transmission in spite of cancellation:
  Corresponding Tx Buffer Transmission Occurred bit MCAN_TXBTO.TOx set
  Corresponding Tx Buffer Cancellation Finished bit MCAN_TXBCF.CFx set
- Arbitration lost or frame transmission disturbed:
  Corresponding Tx Buffer Transmission Occurred bit MCAN_TXBTO.TOx not set
  Corresponding Tx Buffer Cancellation Finished bit MCAN_TXBCF.CFx set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx Event FIFO element is written with Event Type ET = "10" (transmission in spite of cancellation).

47.5.1.8 Power Down (Sleep Mode)

The MCAN can be set into power down mode via bit MCAN_CCCR.CSR.

When all pending transmission requests have completed, the MCAN waits until bus idle state is detected. Then the MCAN sets MCAN_CCCR.INIT to prevent any further CAN transfers. Now the MCAN acknowledges that it is ready for power down by setting to one the bit MCAN_CCCR.CSA. In this state, before the clocks are switched off, further register accesses can be made. A write access to MCAN_CCCR.INIT will have no effect. Now the bus clock (peripheral clock) and the CAN core clock may be switched off.

To leave Power-down mode, the application has to turn on the MCAN clocks before clearing CC Control Register flag MCAN_CCCR.CSR. The MCAN will acknowledge this by clearing MCAN_CCCR.CSA. The application can then restart CAN communication by clearing the bit CCCR.INIT.

Downloaded from Arrow.com.

47.5.1.9 Test Modes

To enable write access to the MCAN Test register (MCAN_TEST) (see Section 47.6.5), bit MCAN_CCCR.TEST must be set. This allows the configuration of the test modes and test functions.

Four output functions are available for the CAN transmit pin CANTX by programming MCAN_TEST.TX. Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor the MCAN's bit timing and it can drive constant dominant or recessive values. The actual value at pin CANRX can be read from MCAN_TEST.RX. Both functions can be used to check the CAN bus' physical layer.

Due to the synchronization mechanism between CAN clock and system bus clock domain, there may be a delay of several system bus clock periods between writing to MCAN_TEST.TX until the new configuration is visible at output pin CANTX. This applies also when reading input pin CANRX via MCAN_TEST.RX.

Note: Test modes should be used for production tests or self-test only. The software control for pin CANTX interferes with all CAN protocol functions. It is not recommended to use test modes for application.

External Loop Back Mode

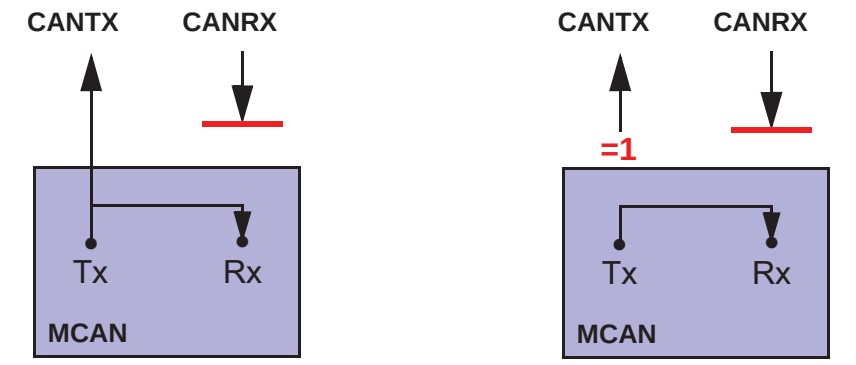
The MCAN can be set in External Loop Back mode by setting the bit MCAN_TEST.LBCK. In Loop Back mode, the MCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into an Rx Buffer or an Rx FIFO. Figure 47-4 shows the connection of signals CANTX and CANRX to the MCAN in External Loop Back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the MCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loop Back mode. In this mode the MCAN performs an internal feedback from its Tx output to its Rx input. The actual value of the CANRX input pin is disregarded by the MCAN. The transmitted messages can be monitored at the CANTX pin.

Internal Loop Back Mode

Internal Loop Back mode is entered by setting bits MCAN_TEST.LBCK and MCAN_CCCR.MON. This mode can be used for a "Hot Selftest", meaning the MCAN can be tested without affecting a running CAN system connected to the pins CANTX and CANRX. In this mode pin CANRX is disconnected from the MCAN and pin CANTX is held recessive. Figure 47-4 shows the connection of CANTX and CANRX to the MCAN in case of Internal Loop Back mode.

Figure 47-4. Pin Control in Loop Back Modes



External Loop Back Mode Internal Loop Back Mode

Downloaded from Arrow.com.

47.5.2 Timestamp Generation

For timestamp generation the MCAN supplies a 16-bit wrap-around counter. A prescaler TSCC.TCP can be configured to clock the counter in multiples of CAN bit times (1...16). The counter is readable via MCAN_TSCV.TSC. A write access to the Timestamp Counter Value register (MCAN_TSCV) resets the counter to zero. When the timestamp counter wraps around interrupt flag IR.TSW is set.

On start of frame reception / transmission the counter value is captured and stored into the timestamp section of an Rx Buffer / Rx FIFO (RXTS[15:0]) or Tx Event FIFO (TXTS[15:0]) element.

By programming bit MCAN_TSCC.TSS an external 16-bit timestamp can be used.

47.5.3 Timeout Counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO, the MCAN supplies a 16-bit Timeout Counter. It operates as down-counter and uses the same prescaler controlled by TSCC.TCP as the Timestamp Counter. The Timeout Counter is configured via the Timeout Counter Configuration register (MCAN_TOCC). The actual counter value can be read from MCAN_TOCV.TOC. The Timeout Counter can only be started while MCAN_CCCR.INIT = '0'. It is stopped when MCAN_CCCR.INIT = '1', e.g. when the MCAN enters Bus_Off state.

The operation mode is selected by MCAN_TOCC.TOS. When operating in Continuous mode, the counter starts when MCAN_CCCR.INIT is reset. A write to MCAN_TOCV presets the counter to the value configured by MCAN_TOCC.TOP and continues down-counting.

When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by MCAN_TOCC.TOP. Down-counting is started when the first FIFO element is stored. Writing to MCAN_TOCV has no effect.

When the counter reaches zero, interrupt flag MCAN_IR.TOO is set. In Continuous mode, the counter is immediately restarted at MCAN_TOCC.TOP.

Note: The clock signal for the Timeout Counter is derived from the CAN Core's sample point signal. Therefore the point in time where the Timeout Counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN Core. If the baud rate switch feature in CAN FD is used, the timeout counter is clocked differently in arbitration and data field.

47.5.4 Rx Handling

The Rx Handler controls the acceptance filtering, the transfer of received messages to the Rx Buffers or to one of the two Rx FIFOs, as well as the Rx FIFO's Put and Get Indices.

47.5.4.1 Acceptance Filtering

The MCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to an Rx Buffer or to Rx FIFO 0,1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - range filter (from - to)
  - filter for one or two dedicated IDs
  - classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled / disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

Downloaded from Arrow.com.

- Global Filter Configuration (MCAN_GFC)
- Standard ID Filter Configuration (MCAN_SIDFC)
- Extended ID Filter Configuration (MCAN_XIDFC)
- Extended ID and Mask (MCAN_XIDAM)

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx Buffer
- Store received frame in Rx Buffer and generate pulse at filter event pin
- Reject received frame
- Set High Priority Message interrupt flag (MCAN_IR.HPM)
- Set High Priority Message interrupt flag (MCAN_IR.HPM) and store received frame in FIFO 0 or FIFO 1

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx Buffer or Rx FIFO has been found, the Message Handler starts writing the received message data in portions of 32 bit to the matching Rx Buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (e.g. CRC error), this message is discarded with the following impact on the effected Rx Buffer or Rx FIFO:

- Rx Buffer
  New Data flag of matching Rx Buffer is not set, but Rx Buffer (partly) overwritten with received data. For error type, see MCAN_PSR.LEC and MCAN_PSR.FLEC.
- Rx FIFO
  Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data. For error type, see MCAN_PSR.LEC and MCAN_PSR.FLEC. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in Rx FIFO Overwrite Mode have to be considered.

Note: When an accepted message is written to one of the two Rx FIFOs, or into an Rx Buffer, the unmodified received identifier is stored independent of the filter(s) used. The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

Range Filter

The filter matches for all received frames with Message IDs in the range defined by SF1ID/SF2ID resp. EF1ID/EF2ID.

There are two possibilities when range filtering is used together with extended frames:

EFT = “00”: The Message ID of received frames is ANDed with MCAN_XIDAM before the range filter is applied.

EFT = “11”: MCAN_XIDAM is not used for range filtering.

Filter for Specific IDs

A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with SF1ID = SF2ID resp. EF1ID = EF2ID.

Classic Bit Mask Filter

Classic bit mask filtering is intended to filter groups of Message IDs by masking single bits of a received Message ID. With classic bit mask filtering SF1ID/EF1ID is used as Message ID filter, while SF2ID/EF2ID is used as filter mask.

A zero bit at the filter mask will mask out the corresponding bit position of the configured ID filter, e.g. the value of the received Message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received Message ID where the corresponding mask bits are one are relevant for acceptance filtering.

In case all mask bits are one, a match occurs only when the received Message ID and the Message ID filter are identical. If all mask bits are zero, all Message IDs match.

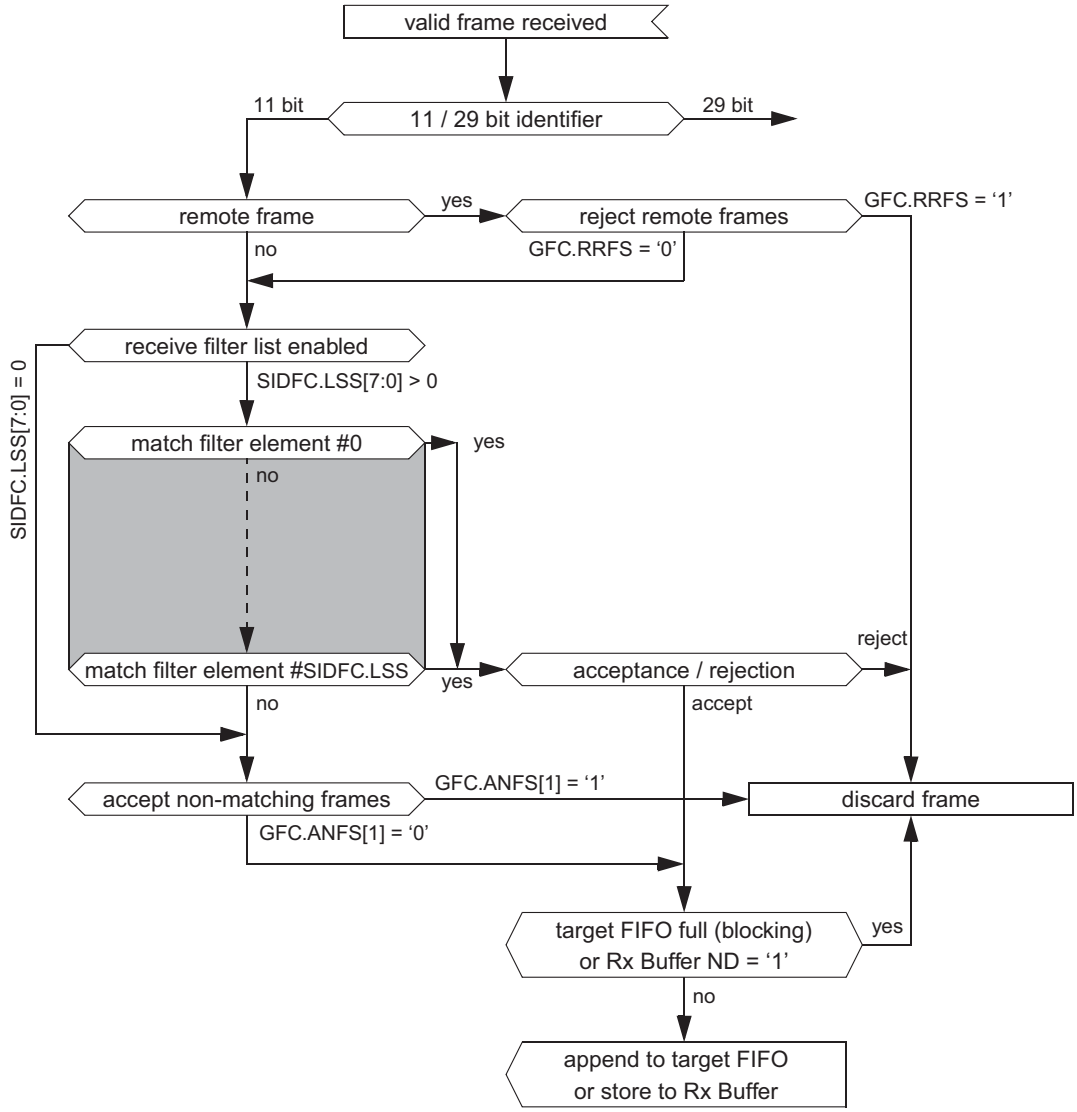
Downloaded from Arrow.com.

Standard Message ID Filtering

Figure 47-5 below shows the flow for standard Message ID (11-bit Identifier) filtering. The Standard Message ID Filter element is described in Section 47.5.7.5.

Controlled by MCAN_GFC and MCAN_SIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

Figure 47-5. Standard Message ID Filter Path



Downloaded from Arrow.com.

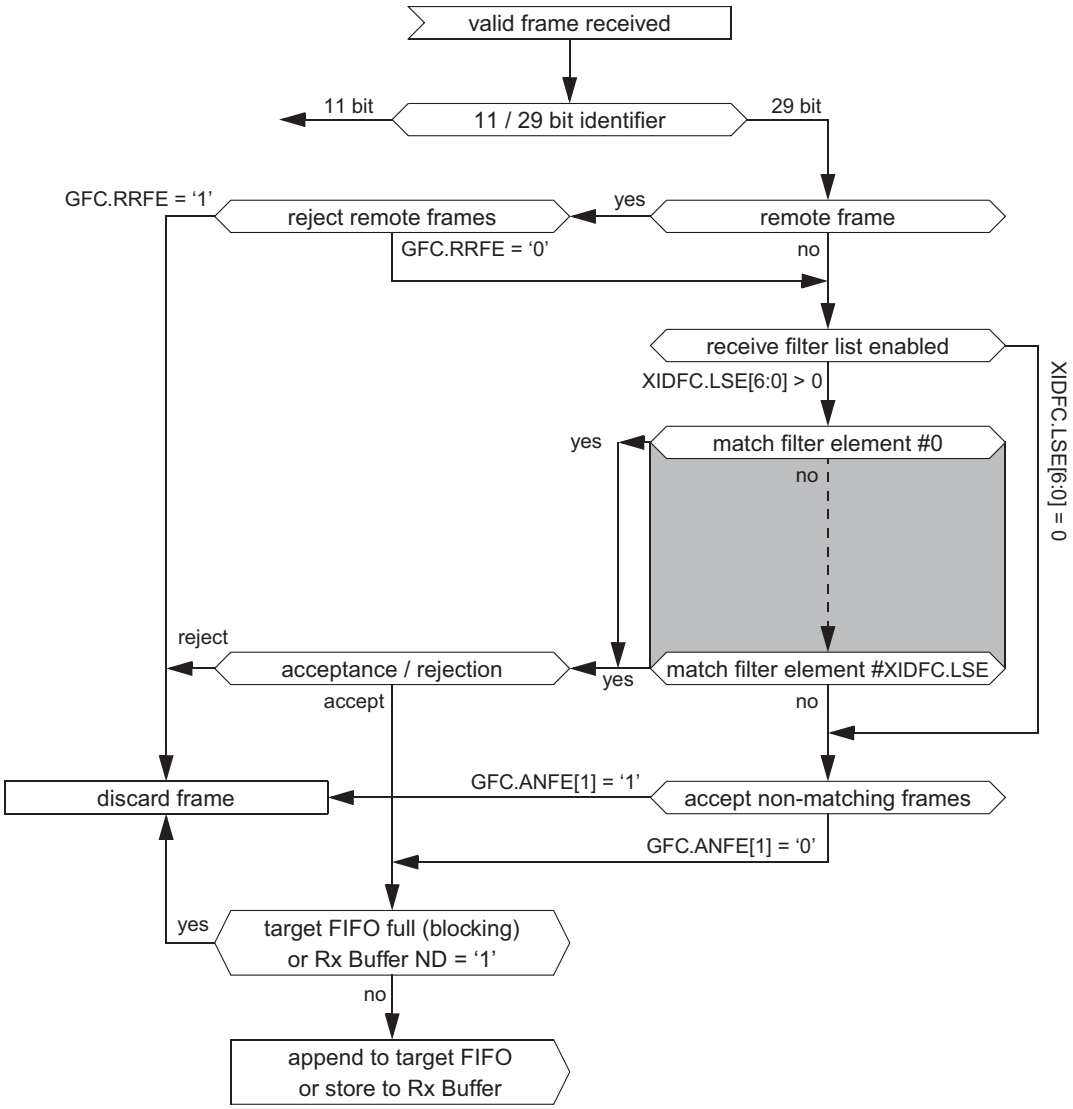
Extended Message ID Filtering

Figure 47-6 below shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID Filter element is described in Section 47.5.7.6.

Controlled by MCAN_GFC and MCAN_XIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

MCAN_XIDAM is ANDed with the received identifier before the filter list is executed.

Figure 47-6. Extended Message ID Filter Path



Downloaded from Arrow.com.

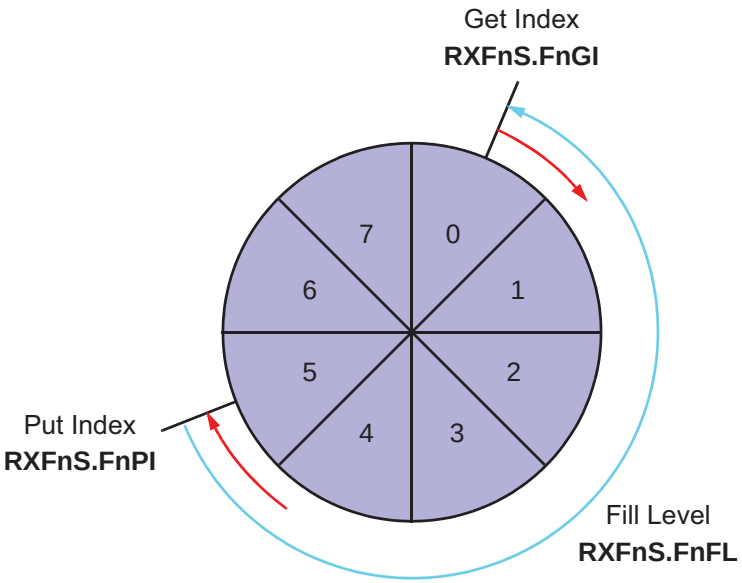
47.5.4.2 Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via the Rx FIFO 0 Configuration register (MCAN_RXF0C) and the Rx FIFO 1 Configuration register (MCAN_RXF1C).

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1, see Section 47.5.4.1. The Rx FIFO element is described in Section 47.5.7.2.

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by MCAN_RXFnC.FnWM, interrupt flag MCAN_IR.RFnW is set. When the Rx FIFO Put Index reaches the Rx FIFO Get Index, an Rx FIFO Full condition is signalled by MCAN_RXFnS.FnF. In addition, the interrupt flag MCAN_IR.RFnF is set.

Figure 47-7. Rx FIFO Status



When reading from an Rx FIFO, Rx FIFO Get Index RXFnS.FnGI × FIFO Element Size has to be added to the corresponding Rx FIFO start address RXFnC.FnSA.

Table 47-4. Rx Buffer / FIFO Element Size

RXESC.RBDS[2:0] RXESC.FnDS[2:0]	Data Field [bytes]	FIFO Element Size [RAM words]
0	8	4
1	12	5
2	16	6
3	20	7
4	24	8
5	32	10
6	48	14
7	64	18

Downloaded from Arrow.com.

Rx FIFO Blocking Mode

The Rx FIFO blocking mode is configured by MCAN_RXFnC.FnOM = ‘0’. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (MCAN_RXFnS.FnPI = MCAN_RXFnS.FnGI), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. An Rx FIFO full condition is signalled by MCAN_RXFnS.FnF = ‘1’. In addition, the interrupt flag MCAN_IR.RFnF is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signalled by MCAN_RXFnS.RFnL = ‘1’. In addition, the interrupt flag MCAN_IR.RFnL is set.

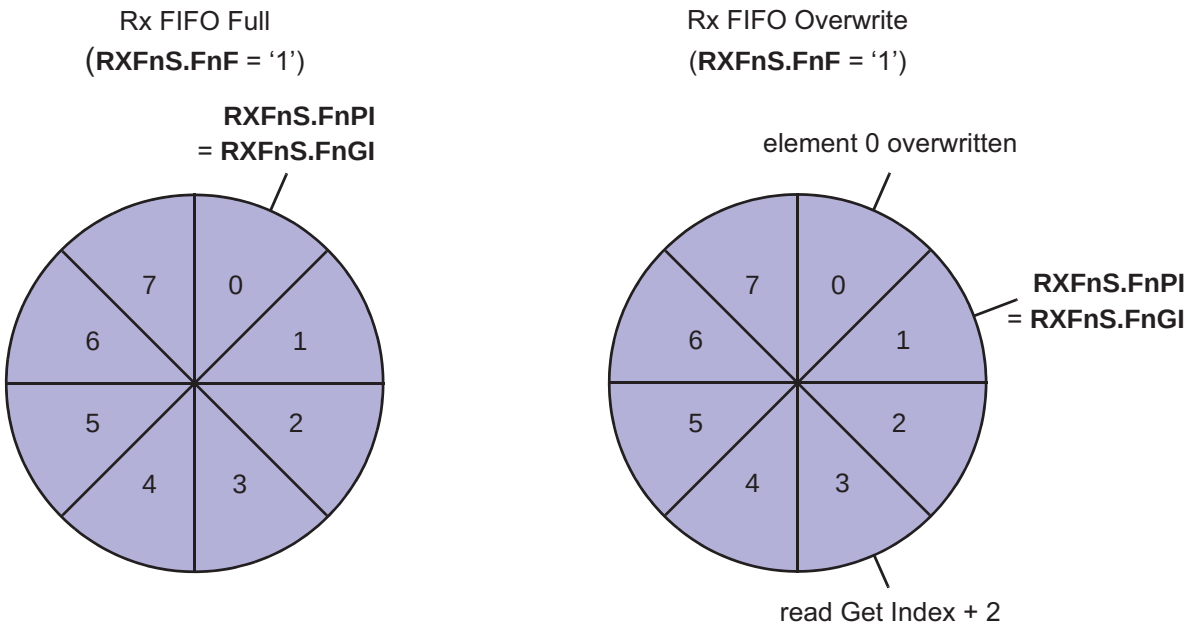
Rx FIFO Overwrite Mode

The Rx FIFO overwrite mode is configured by MCAN_RXFnC.FnOM = ‘1’.

When an Rx FIFO full condition (MCAN_RXFnS.FnPI = MCAN_RXFnS.FnGI) is signalled by MCAN_RXFnS.FnF = ‘1’, the next message accepted for the FIFO will overwrite the oldest FIFO message. Put and get index are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signalled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for that is, that it might happen, that a received message is written to the Message RAM (put index) while the processor is reading from the Message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the processor accesses the Rx FIFO. Figure 47-8 shows an offset of two with respect to the get index when reading the Rx FIFO. In this case the two messages stored in element 1 and 2 are lost.

Figure 47-8. Rx FIFO Overflow Handling



After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO Acknowledge Index RXFnA.FnA. This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (MCAN_RXFnS.FnF = ‘0’).

Downloaded from Arrow.com.

47.5.4.3 Dedicated Rx Buffers

The MCAN supports up to 64 dedicated Rx Buffers. The start address of the dedicated Rx Buffer section is configured via MCAN_RXBC.RBSA.

For each Rx Buffer, a Standard or Extended Message ID Filter Element with SFEC / EFEC = 7 and SFID2 / EFID2[10:9] = 0 has to be configured (see Section 47.5.7.5 and Section 47.5.7.6).

After a received message has been accepted by a filter element, the message is stored into the Rx Buffer in the Message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition, the flag MCAN_IR.DRX (Message stored in dedicated Rx Buffer) in MCAN_IR is set.

Table 47-5. Example Filter Configuration for Rx Buffers

Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID message 1	0	0
1	ID message 2	0	1
2	ID message 3	0	2

After the last word of a matching received message has been written to the Message RAM, the respective New Data flag in the New Data 1 register (MCAN_NDAT1) and New Data 2 register (MCAN_NDAT2) is set. As long as the New Data flag is set, the respective Rx Buffer is locked against updates from received matching frames. The New Data flags have to be reset by the processor by writing a '1' to the respective bit position.

While an Rx Buffer's New Data flag is set, a Message ID Filter Element referencing this specific Rx Buffer will not match, causing the acceptance filtering to continue. Following Message ID Filter Elements may cause the received message to be stored into another Rx Buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

Rx Buffer Handling

- Reset interrupt flag IR.DRX
- Read New Data registers
- Read messages from Message RAM
- Reset New Data flags of processed messages

47.5.4.4 Debug on CAN Support

Debug messages are stored into Rx Buffers. For debug handling three consecutive Rx buffers (e.g. #61, #62, #63) have to be used for storage of debug messages A, B, and C. The format is the same as for an Rx Buffer or an Rx FIFO element (see Section 47.5.7.2 "Rx Buffer and FIFO Element").

Advantage: Fixed start address for the DMA transfers (relative to MCAN_RXBC.RBSA), no additional configuration required.

For filtering of debug messages Standard / Extended Filter Elements with SFEC / EFEC = 7 have to be set up. Messages matching these filter elements are stored into the Rx Buffers addressed by SFID2 / EFID2[5:0].

After message C has been stored, the DMA request output m_can_dma_req is activated and the three messages can be read from the Message RAM under DMA control. The RAM words holding the debug messages will not be changed by the MCAN while m_can_dma_req is activated. The behavior is similar to that of an Rx Buffer with its New Data flag set.

After the DMA has completed, the MCAN is prepared to receive the next set of debug messages.

Filtering for Debug Messages

Filtering for debug messages is done by configuring one Standard / Extended Message ID Filter Element for each of the three debug messages. To enable a filter element to filter for debug messages SFEC / EFEC has to be

Downloaded from Arrow.com.

programmed to “111”. In this case fields SFID1 / SFID2 and EFID1 / EFID2 have a different meaning (see Section 47.5.7.5 and Section 47.5.7.6). While SFID2 / EFID2[10:9] controls the debug message handling state machine, SFID2 / EFID2[5:0] controls the location for storage of a received debug message.

When a debug message is stored, neither the respective New Data flag nor MCAN_IR.DRX are set. The reception of debug messages can be monitored via RXF1S.DMS.

Table 47-6. Example Filter Configuration for Debug Messages

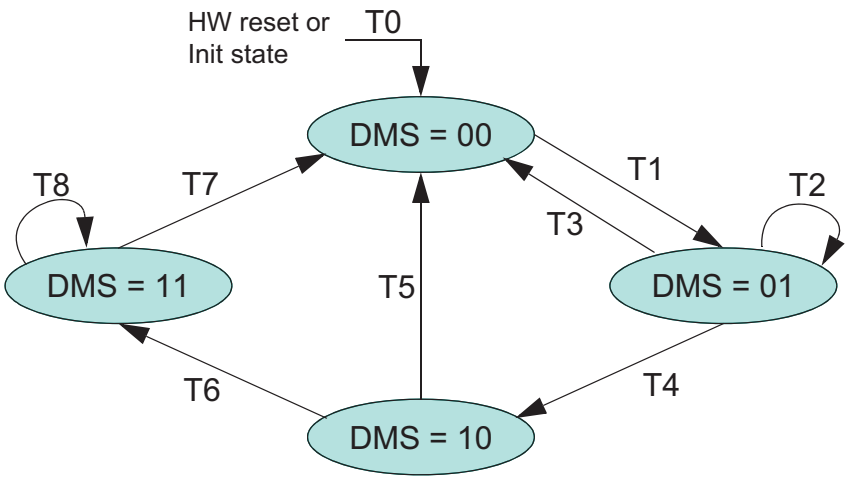
Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message A	1	11 1101
1	ID debug message B	2	11 1110
2	ID debug message C	3	11 1111

Debug Message Handling

The debug message handling state machine assures that debug messages are stored to three consecutive Rx Buffers in the correct order. In case of missing messages the process is restarted. The DMA request is activated only when all three debug messages A, B, C have been received in the correct order.

The status of the debug message handling state machine is signalled via MCAN_RXF1S.DMS.

Figure 47-9. Debug Message Handling State Machine



T0: reset m_cam_dma_req output, enable reception of debug messages A, B, and C

T1: reception of debug message A

T2: reception of debug message A

T3: reception of debug message C

T4: reception of debug message B

T5: reception of debug messages A, B

T6: reception of debug message C

T7: DMA transfer completed

T8: reception of debug message A,B,C (message rejected)

Downloaded from Arrow.com.

47.5.5 Tx Handling

Note: The Tx Handler handles transmission requests for the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue. It controls the transfer of transmit messages to the CAN Core, the Put and Get Indices, and the Tx Event FIFO. Up to 32 Tx Buffers can be set up for message transmission. The Tx Buffer element is described in Section 47.5.7.3. AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation.

The Tx Handler starts a Tx scan to check for the highest priority pending Tx request (Tx Buffer with lowest Message ID) when MCAN_TXBRP is updated, or when a transmission has been started.

47.5.5.1 Transmit Pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU's CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If e.g. CAN ECU-1 has the transmit pause feature enabled and is requested by its application software to transmit four messages, it will, after the first successful message transmission, wait for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The transmit pause feature is controlled by bit MCAN_CCCR.TXP. If the bit is set, the MCAN will, each time it has successfully transmitted a message, pause for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is transmit pause disabled (MCAN_CCCR.TXP = '0').

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

47.5.5.2 Dedicated Tx Buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the processor. Each dedicated Tx Buffer is configured with a specific Message ID. In case that multiple Tx Buffers are configured with the same Message ID, the Tx Buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an Add Request via MCAN_TXBAR.ARn. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx Queue and externally with messages on the CAN bus, and are sent out according to their Message ID.

A dedicated Tx Buffer allocates Element Size 32-bit words in the Message RAM (see Table 47-7). Therefore the start address of a dedicated Tx Buffer in the Message RAM is calculated by adding transmit buffer index (0…31) × Element Size to the Tx Buffer Start Address TXBC.TBSA.

Table 47-7. Tx Buffer / FIFO / Queue Element Size

TXESC.TBDS[2:0]	Data Field [bytes]	Element Size [RAM words]
0	8	4
1	12	5
2	16	6
3	20	7
4	24	8

Downloaded from Arrow.com.

Table 47-7. Tx Buffer / FIFO / Queue Element Size

TXESC.TBDS[2:0]	Data Field [bytes]	Element Size [RAM words]
5	32	10
6	48	14
7	64	18

47.5.5.3 Tx FIFO

Tx FIFO operation is configured by programming MCAN_TXBC.TFQM to '0'. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the Get Index MCAN_TXFQS.TFGI. After each transmission the Get Index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same Message ID from different Tx Buffers in the order these messages have been written to the Tx FIFO. The MCAN calculates the Tx FIFO Free Level MCAN_TXFQS.TFFL as difference between Get and Put Index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index MCAN_TXFQS.TFQPI. An Add Request increments the Put Index to the next free Tx FIFO element. When the Put Index reaches the Get Index, Tx FIFO Full (MCAN_TXFQS.TFQF = '1') is signalled. In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a '1' to the TXBAR bit related to the Tx Buffer referenced by the Tx FIFO's Put Index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx Buffers starting with the Put Index. The transmissions are then requested via MCAN_TXBAR. The Put Index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx Buffers as indicated by the Tx FIFO Free Level.

When a transmission request for the Tx Buffer referenced by the Get Index is cancelled, the Get Index is incremented to the next Tx Buffer with pending transmission request and the Tx FIFO Free Level is recalculated. When transmission cancellation is applied to any other Tx Buffer, the Get Index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates Element Size 32-bit words in the Message RAM (see Table 47-7). Therefore the start address of the next available (free) Tx FIFO Buffer is calculated by adding Tx FIFO/Queue Put Index MCAN_TXFQS.TFQPI (0…31) × Element Size to the Tx Buffer Start Address MCAN_TXBC.TBSA.

47.5.5.4 Tx Queue

Tx Queue operation is configured by programming MCAN_TXBC.TFQM to '1'. Messages stored in the Tx Queue are transmitted starting with the message with the lowest Message ID (highest priority). In case that multiple Queue Buffers are configured with the same Message ID, the Queue Buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx Buffer referenced by the Put Index MCAN_TXFQS.TFQPI. An Add Request cyclically increments the Put Index to the next free Tx Buffer. In case that the Tx Queue is full (MCAN_TXFQS.TFQF = '1'), the Put Index is not valid and no further message should be written to the Tx Queue until at least one of the requested messages has been sent out or a pending transmission request has been cancelled.

The application may use register MCAN_TXBRP instead of the Put Index and may place messages to any Tx Buffer without pending transmission request.

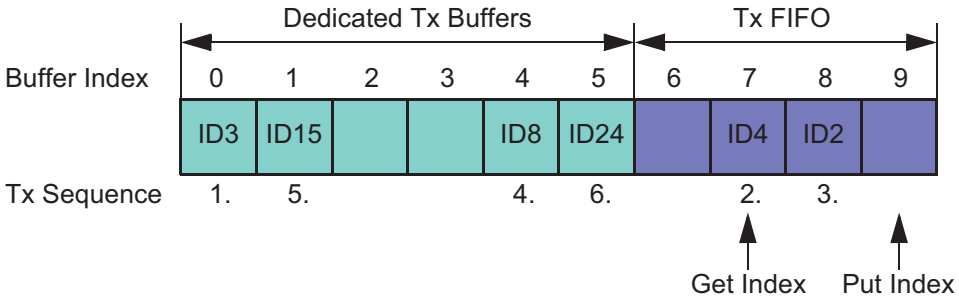
Downloaded from Arrow.com.

A Tx Queue Buffer allocates Element Size 32-bit words in the Message RAM (see Table 47-7). Therefore the start address of the next available (free) Tx Queue Buffer is calculated by adding Tx FIFO/Queue Put Index MCAN_TXFQS.TFQPI (0…31) × Element Size to the Tx Buffer Start Address MCAN_TXBC.TBSA.

47.5.5.5 Mixed Dedicated Tx Buffers / Tx FIFO

In this case the Tx Buffers section in the Message RAM is subdivided into a set of dedicated Tx Buffers and a Tx FIFO. The number of dedicated Tx Buffers is configured by MCAN_TXBC.NDTB. The number of Tx Buffers assigned to the Tx FIFO is configured by MCAN_TXBC.TFQS. In case MCAN_TXBC.TFQS is programmed to zero, only dedicated Tx Buffers are used.

Figure 47-10. Example of Mixed Configuration Dedicated Tx Buffers / Tx FIFO



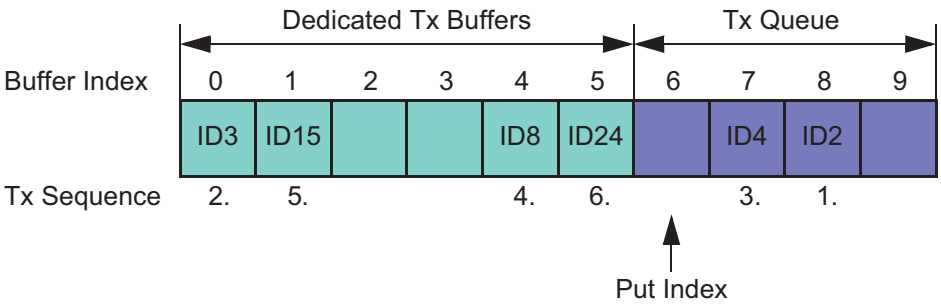
Tx prioritization:

- Scan dedicated Tx Buffers and oldest pending Tx FIFO Buffer (referenced by MCAN_TXFS.TFGI)
- Buffer with lowest Message ID gets highest priority and is transmitted next

47.5.5.6 Mixed Dedicated Tx Buffers / Tx Queue

In this case the Tx Buffers section in the Message RAM is subdivided into a set of dedicated Tx Buffers and a Tx Queue. The number of dedicated Tx Buffers is configured by MCAN_TXBC.NDTB. The number of Tx Queue Buffers is configured by MCAN_TXBC.TFQS. In case MCAN_TXBC.TFQS is programmed to zero, only dedicated Tx Buffers are used.

Figure 47-11. Example of Mixed Configuration Dedicated Tx Buffers / Tx Queue



Tx prioritization:

- Scan all Tx Buffers with activated transmission request
- Tx Buffer with lowest Message ID gets highest priority and is transmitted next

47.5.5.7 Transmit Cancellation

The MCAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR-based applications. To cancel a requested transmission from a dedicated Tx Buffer or a Tx Queue Buffer, the processor has to write a ‘1’ to the corresponding bit position (=number of Tx Buffer) of register MCAN_TXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Downloaded from Arrow.com.

Successful cancellation is signalled by setting the corresponding bit of register MCAN_TXBCF to ‘1’.

In case a transmit cancellation is requested while a transmission from a Tx Buffer is already ongoing, the corresponding TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding MCAN_TXBTO and MCAN_TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding MCAN_TXBCF bit is set.

Note: In case a pending transmission is cancelled immediately before this transmission could have been started, there follows a short time window where no transmission is started even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.

47.5.5.8 Tx Event Handling

To support Tx event handling the MCAN has implemented a Tx Event FIFO. After the MCAN has transmitted a message on the CAN bus, Message ID and timestamp are stored in a Tx Event FIFO element. To link a Tx event to a Tx Event FIFO element, the Message Marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element.

The Tx Event FIFO can be configured to a maximum of 32 elements. The Tx Event FIFO element is described in Section 47.5.4.4.

When a Tx Event FIFO full condition is signalled by IR.TEFF, no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx Event FIFO is full, this event is discarded and interrupt flag MCAN_IR.TEFL is set.

To avoid a Tx Event FIFO overflow, the Tx Event FIFO watermark can be used. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by MCAN_TXEFC.EFWM, interrupt flag MCAN_IR.TEFW is set.

When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index MCAN_TXEFS.EFGI has to be added to the Tx Event FIFO start address MCAN_TXEFC.EFSA.

47.5.6 FIFO Acknowledge Handling

The Get Indices of Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index (see Section 47.6.28, Section 47.6.32, and Section 47.6.46). Writing to the FIFO Acknowledge Index will set the FIFO Get Index to the FIFO Acknowledge Index plus one and thereby updates the FIFO Fill Level. There are two use cases:

When only a single element has been read from the FIFO (the one being pointed to by the Get Index), this Get Index value is written to the FIFO Acknowledge Index.

When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO Acknowledge Index only once at the end of that read sequence (value: Index of the last element read), to update the FIFO’s Get Index.

Due to the fact that the processor has free access to the MCAN’s Message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (Get Index not considered). This might be useful when reading a High Priority Message from one of the two Rx FIFOs. In this case the FIFO’s Acknowledge Index should not be written because this would set the Get Index to a wrong position and also alters the FIFO’s Fill Level. In this case some of the older FIFO elements would be lost.

Note: The application has to ensure that a valid value is written to the FIFO Acknowledge Index. The MCAN does not check for erroneous values.

Downloaded from Arrow.com.

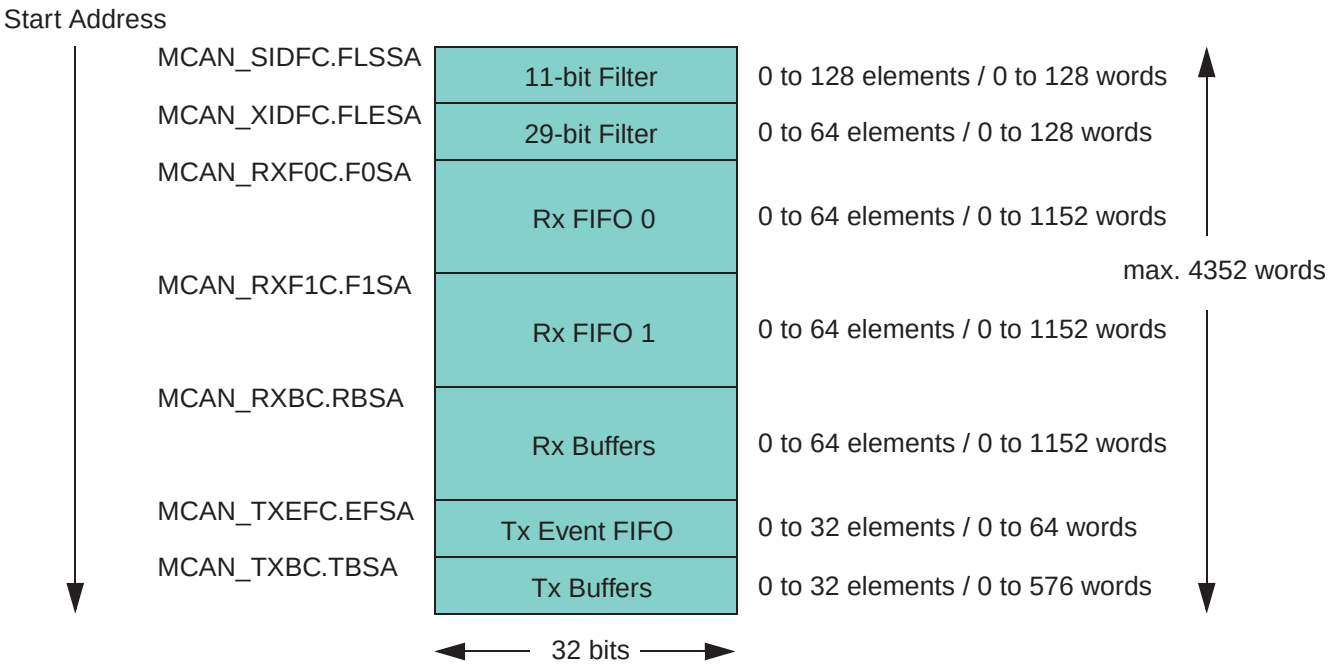
47.5.7 Message RAM

47.5.7.1 Message RAM Configuration

The Message RAM has a width of 32 bits. The MCAN module can be configured to allocate up to 4352 words in the Message RAM. It is not necessary to configure each of the sections listed in Figure 47-12, nor is there any restriction with respect to the sequence of the sections.

When operated in CAN FD mode, the required Message RAM size depends on the element size configured for Rx FIFO0, Rx FIFO1, Rx Buffers, and Tx Buffers via MCAN_RXESC.F0DS, MCAN_RXESC.F1DS, MCAN_RXESC.RBDS, and MCAN_TXESC.TBDS.

Figure 47-12. Message RAM Configuration



When the MCAN addresses the Message RAM, it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses; i.e., only bits 15 to 2 are evaluated, the two least significant bits are ignored.

Note: The MCAN does not check for erroneous configuration of the Message RAM. The configuration of the start addresses of the different sections and the number of elements of each section must be checked carefully to avoid falsification or loss of data.

Downloaded from Arrow.com.

47.5.7.2 Rx Buffer and FIFO Element

Up to 64 Rx Buffers and two Rx FIFOs can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx Buffer / FIFO element is shown in Table 47-8 below. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register MCAN_RXESC.

Table 47-8. Rx Buffer and FIFO Element

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R0	ESI	XTD	RTR	ID[28:0]																												
R1	ANMF	FIDX[6:0]							–				EDL	BRS	DLC[3:0]		RXTS[15:0]															
R2	DB3[7:0]								DB2[7:0]								DB1[7:0]								DB0[7:0]							
R3	DB7[7:0]								DB6[7:0]								DB5[7:0]								DB4[7:0]							
...	...								...								...								...							
Rn	DBm[7:0]								DBm-1[7:0]								DBm-2[7:0]								DBm-3[7:0]							

- **R0 Bit 31 ESI: Error State Indicator**

0: Transmitting node is error active.

1: Transmitting node is error passive.

- **R0 Bit 30 XTD: Extended Identifier**

Signals to the processor whether the received frame has a standard or extended identifier.

0: 11-bit standard identifier.

1: 29-bit extended identifier.

- **R0 Bit 29 RTR: Remote Transmission Request**

Signals to the processor whether the received frame is a data frame or a remote frame.

0: Received frame is a data frame.

1: Received frame is a remote frame.

Note: There are no remote frames in CAN FD format. In case a CAN FD frame was received (EDL = 1), bit RTR reflects the state of the reserved bit r1.

- **R0 Bits 28:0 ID[28:0]: Identifier**

Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].

- **R1 Bit 31 ANMF: Accepted Non-matching Frame**

Acceptance of non-matching frames may be enabled via MCAN_GFC.ANFS and MCAN_GFC.ANFE.

0: Received frame matching filter index FIDX.

1: Received frame did not match any Rx filter element.

- **R1 Bits 30:24 FIDX[6:0]: Filter Index**

0-127: Index of matching Rx acceptance filter element (invalid if ANMF = '1').
Range is 0 to MCAN_SIDFC.LSS - 1 resp. MCAN_XIDFC.LSE - 1.

Downloaded from Arrow.com.

- **R1 Bit 21 EDL: Extended Data Length**

0: Standard frame format.

1: CAN FD frame format (new DLC-coding and CRC).

- **R1 Bit 20 BRS: Bit Rate Switch**

0: Frame received without bit rate switching.

1: Frame received with bit rate switching.

- **R1 Bits 19:16 DLC[3:0]: Data Length Code**

0-8: CAN + CAN FD: received frame has 0-8 data bytes.

9-15: CAN: received frame has 8 data bytes.

9-15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes.

- **R1 Bits 15:0 RXTS[15:0]: Rx Timestamp**

Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler MCAN_TSCC.TCP.

- **R2 Bits 31:24 DB3[7:0]: Data Byte 3**
- **R2 Bits 23:16 DB2[7:0]: Data Byte 2**
- **R2 Bits 15:8 DB1[7:0]: Data Byte 1**
- **R2 Bits 7:0 DB0[7:0]: Data Byte 0**
- **R3 Bits 31:24 DB7[7:0]: Data Byte 7**
- **R3 Bits 23:16 DB6[7:0]: Data Byte 6**
- **R3 Bits 15:8 DB5[7:0]: Data Byte 5**
- **R3 Bits 7:0 DB4[7:0]: Data Byte 4**

… … …

- **Rn Bits 31:24 DBm[7:0]: Data Byte m**
- **Rn Bits 23:16 DBm-1[7:0]: Data Byte m-1**
- **Rn Bits 15:8 DBm-2[7:0]: Data Byte m-2**
- **Rn Bits 7:0 DBm-3[7:0]: Data Byte m-3**

Note: Depending on the configuration of the element size (MCAN_RXESC), between two and sixteen 32-bit words (Rn = 3 ..17) are used for storage of a CAN message's data field.

Downloaded from Arrow.com.

47.5.7.3 Tx Buffer Element

The Tx Buffers section can be configured to hold dedicated Tx Buffers as well as a Tx FIFO / Tx Queue. In case that the Tx Buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx Queue, the dedicated Tx Buffers start at the beginning of the Tx Buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler distinguishes between dedicated Tx Buffers and Tx FIFO / Tx Queue by evaluating the Tx Buffer configuration TXBC.TFQS and TXBC.NDTB. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register TXESC.

Table 47-9. Tx Buffer Element

	31	30	29	28:24	23	22	21:20	19:16	15:8	7:0
T0	reserved	XTD	RTR	ID[28:0]						
T1	MM[7:0]				EFC	reserved		DLC[3:0]	reserved	
T2	DB3[7:0]				DB2[7:0]				DB1[7:0]	DB0[7:0]
T3	DB7[7:0]				DB6[7:0]				DB5[7:0]	DB4[7:0]
...	...				...				...	...
Tn	DBm[7:0]				DBm-1[7:0]				DBm-2[7:0]	DBm-3[7:0]

- **T0 Bit 30 XTD: Extended Identifier**

0: 11-bit standard identifier.

1: 29-bit extended identifier.

- **T0 Bit 29 RTR: Remote Transmission Request**

0: Transmit data frame.

1: Transmit remote frame.

Note: When RTR = 1, the MCAN transmits a remote frame according to ISO11898-1, even if MCAN_CCCR.CME enables the transmission in CAN FD format.

- **T0 Bits 28:0 ID[28:0]: Identifier**

Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].

- **T1 Bits 31:24 MM[7:0]: Message Marker**

Written by processor during Tx Buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status.

- **T1 Bit 23 EFC: Event FIFO Control**

0: Do not store Tx events.

1: Store Tx events.

- **T1 Bits 19:16 DLC[3:0]: Data Length Code**

0-8: CAN + CAN FD: transmit frame has 0-8 data bytes.

9-15: CAN: transmit frame has 8 data bytes.

9-15: CAN FD: transmit frame has 12/16/20/24/32/48/64 data bytes.

Downloaded from Arrow.com.

- **T2 Bits 31:24 DB3[7:0]: Data Byte 3**
- **T2 Bits 23:16 DB2[7:0]: Data Byte 2**
- **T2 Bits 15:8 DB1[7:0]: Data Byte 1**
- **T2 Bits 7:0 DB0[7:0]: Data Byte 0**
- **T3 Bits 31:24 DB7[7:0]: Data Byte 7**
- **T3 Bits 23:16 DB6[7:0]: Data Byte 6**
- **T3 Bits 15:8 DB5[7:0]: Data Byte 5**
- **T3 Bits 7:0 DB4[7:0]: Data Byte 4**

… … …

- **Tn Bits 31:24 DBm[7:0]: Data Byte m**
- **Tn Bits 23:16 DBm-1[7:0]: Data Byte m-1**
- **Tn Bits 15:8 DBm-2[7:0]: Data Byte m-2**
- **Tn Bits 7:0 DBm-3[7:0]: Data Byte m-3**

Note: Depending on the configuration of the element size (MCAN_TXESC), between two and sixteen 32-bit words (Tn = 3 ..17) are used for storage of a CAN message's data field.

Downloaded from Arrow.com.

47.5.7.4 Tx Event FIFO Element

Each element stores information about transmitted messages. By reading the Tx Event FIFO the processor gets this information in the order the messages were transmitted. Status information about the Tx Event FIFO can be obtained from register TXEFS.

Table 47-10. Tx Event FIFO Element

	31							24	23							16	15							8	7							0
E0	ESI			XTD			RTR		ID[28:0]																							
E1	MM[7:0]								ET [1:0]		EDL	BRS	DLC[3:0]				TXTS[15:0]															

- **E0 Bit 31 ESI: Error State Indicator**

0: Transmitting node is error active.

1: Transmitting node is error passive.

- **E0 Bit 30 XTD: Extended Identifier**

0: 11-bit standard identifier.

1: 29-bit extended identifier.

- **E0 Bit 29 RTR: Remote Transmission Request**

0: Data frame transmitted.

1: Remote frame transmitted.

- **E0 Bits 28:0 ID[28:0]: Identifier**

Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].

- **E1 Bits 31:24 MM[7:0]: Message Marker**

Copied from Tx Buffer into Tx Event FIFO element for identification of Tx message status.

- **E1 Bit 23:22 ET[1:0]: Event Type**

Value	Description
0	Reserved
1	Tx event
2	Transmission in spite of cancellation (always set for transmissions in DAR mode)
3	Reserved

- **E1 Bit 21 EDL: Extended Data Length**

0: Standard frame format.

1: CAN FD frame format (new DLC-coding and CRC).

- **E1 Bit 20 BRS: Bit Rate Switch**

0: Frame transmitted without bit rate switching.

1: Frame transmitted with bit rate switching.

- **E1 Bits 19:16 DLC[3:0]: Data Length Code**

0-8: CAN + CAN FD: frame with 0-8 data bytes transmitted.

9-15: CAN: frame with 8 data bytes transmitted.

9-15: CAN FD: frame with 12/16/20/24/32/48/64 data bytes transmitted

Downloaded from Arrow.com.

- **E1 Bits 15:0 TXTS[15:0]: Tx Timestamp**

Timestamp Counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler MCAN_TSCC.TCP.

Downloaded from Arrow.com.

47.5.7.5 Standard Message ID Filter Element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a Standard Message ID Filter element, its address is the Filter List Standard Start Address MCAN_SIDFC.FLSSA plus the index of the filter element (0…127).

Table 47-11. Standard Message ID Filter Element

	31	24	23	16	15	8	7	0
S0	SFT[1:0]	SFEC [2:0]	SFID1[10:0]		–		SFID2[10:0]	

- **Bits 31:30 SFT[1:0]: Standard Filter Type**

Value	Description
0	Range filter from SF1ID to SF2ID (SF2ID ≥ SF1ID)
1	Dual ID filter for SF1ID or SF2ID
2	Classic filter: SF1ID = filter, SF2ID = mask
3	Reserved

- **Bit 29:27 SFEC[2:0]: Standard Filter Element Configuration**

All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = “100”, “101”, or “110” a match sets interrupt flag MCAN_IR.HPM and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match.

Value	Description
0	Disable filter element
1	Store in Rx FIFO 0 if filter matches
2	Store in Rx FIFO 1 if filter matches
3	Reject ID if filter matches
4	Set priority if filter matches
5	Set priority and store in FIFO 0 if filter matches
6	Set priority and store in FIFO 1 if filter matches
7	Store into Rx Buffer or as debug message, configuration of SFT[1:0] ignored

- **Bits 26:16 SFID1[10:0]: Standard Filter ID 1**

First ID of standard ID filter element.

When filtering for Rx Buffers or for debug messages this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.

- **Bits 10:0 SFID2[10:0]: Standard Filter ID 2**

This field has a different meaning depending on the configuration of SFEC:

- SFEC = “001”...”110”–Second ID of standard ID filter element
- SFEC = “111”–Filter for Rx Buffers or for debug messages

Downloaded from Arrow.com.

SFID2[10:9] decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence.

Value	Description
0	Store message in a Rx buffer
1	Debug Message A
2	Debug Message B
3	Debug Message C

SFID2[5:0] defines the index of the dedicated Rx Buffer element to which a matching message is stored.

Downloaded from Arrow.com.

47.5.7.6 Extended Message ID Filter Element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended Message ID Filter element, its address is the Filter List Extended Start Address MCAN_XIDFC.FLESA plus two times the index of the filter element (0…63).

Table 47-12. Extended Message ID Filter Element

	31	24	23	16	15	8	7	0
F0	EFEC [2:0]		EFID1[28:0]					
F1	EFT[1:0]	–	EFID2[28:0]					

- **F0 Bit 31:29 EFEC[2:0]: Extended Filter Element Configuration**

All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = “100”, “101”, or “110”, a match sets the interrupt flag MCAN_IR.HPM and, if enabled, an interrupt is generated. In this case, register MCAN_HPMS is updated with the status of the priority match.

Value	Description
0	Disable filter element
1	Store in Rx FIFO 0 if filter matches
2	Store in Rx FIFO 1 if filter matches
3	Reject ID if filter matches
4	Set priority if filter matches
5	Set priority and store in FIFO 0 if filter matches
6	Set priority and store in FIFO 1 if filter matches
7	Store into Rx Buffer or as debug message, configuration of EFT[1:0] ignored

- **F0 Bits 28:0 EFID1[28:0]: Extended Filter ID 1**

First ID of extended ID filter element.

When filtering for Rx Buffers or for debug messages this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only MCAN_XIDAM masking mechanism (see Extended Message ID Filtering) is used.

- **F1 Bits 31:30 EFT[1:0]: Extended Filter Type**

Value	Description
0	Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID)
1	Dual ID filter for EF1ID or EF2ID
2	Classic filter: EF1ID = filter, EF2ID = mask
3	Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID), MCAN_XIDAM mask not applied

Downloaded from Arrow.com.

- **F1 Bits 28:0 EFID2[28:0]: Extended Filter ID 2**

This field has a different meaning depending on the configuration of EFEC:

- EFEC = “001”...”110”–Second ID of extended ID filter element
- EFEC = “111”–Filter for Rx Buffers or for debug messages

EFID2[10:9] decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence.

Value	Description
0	Store message in a Rx buffer
1	Debug Message A
2	Debug Message B
3	Debug Message C

EFID2[5:0] defines the index of the dedicated Rx Buffer element to which a matching message is stored.

Downloaded from Arrow.com.

47.5.8 Hardware Reset Description

After hardware reset, the registers of the MCAN hold the reset values listed in Table 47-13. Additionally the Bus_Off state is reset and the output CANTX is set to recessive (HIGH). The value 0x0001 (MCAN_CCCR.INIT = ‘1’) in the CC Control register enables software initialization. The MCAN does not influence the CAN bus until the processor resets MCAN_CCCR.INIT to ‘0’.

Downloaded from Arrow.com.

47.6 Controller Area Network (MCAN) User Interface

Table 47-13. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Core Release Register	MCAN_CREL	Read-only	0xrrrddddd[1]
0x04	Endian Register	MCAN_ENDN	Read-only	0x87654321
0x00-0x04	Reserved	–	–	–
0x08	Customer Register	MCAN_CUST	Read/Write	0
0x0C	Fast Bit Timing and Prescaler Register	MCAN_FBTP	Read/Write	0x00000A33
0x10	Test Register	MCAN_TEST	Read/Write	0x000000x0[2]
0x14	RAM Watchdog Register	MCAN_RWD	Read/Write	0x00000000
0x18	CC Control Register	MCAN_CCCR	Read/Write	0x00000001
0x1C	Bit Timing and Prescaler Register	MCAN_BTP	Read/Write	0x00000A33
0x20	Timestamp Counter Configuration Register	MCAN_TSCC	Read/Write	0x00000000
0x24	Timestamp Counter Value Register	MCAN_TSCV	Read/Write	0x00000000
0x28	Timeout Counter Configuration Register	MCAN_TOCC	Read/Write	0xFFFF0000
0x2C	Timeout Counter Value Register	MCAN_TOCV	Read/Write	0x0000FFFF
0x30-0x3C	Reserved	–	–	–
0x40	Error Counter Register	MCAN_ECR	Read-only	0x00000000
0x44	Protocol Status Register	MCAN_PSR	Read-only	0x00000707
0x48-0x4C	Reserved	–	–	–
0x50	Interrupt Register	MCAN_IR	Read/Write	0x00000000
0x54	Interrupt Enable Register	MCAN_IE	Read/Write	0x00000000
0x58	Interrupt Line Select Register	MCAN_ILS	Read/Write	0x00000000
0x5C	Interrupt Line Enable Register	MCAN_ILE	Read/Write	0x00000000
0x60-0x7C	Reserved	–	–	–
0x80	Global Filter Configuration Register	MCAN_GFC	Read/Write	0x00000000
0x84	Standard ID Filter Configuration Register	MCAN_SIDFC	Read/Write	0x00000000
0x88	Extended ID Filter Configuration Register	MCAN_XIDFC	Read/Write	0x00000000
0x8C	Reserved	–	–	–
0x90	Extended ID AND Mask Register	MCAN_XIDAM	Read/Write	0x1FFFFFFF
0x94	High Priority Message Status Register	MCAN_HPMS	Read-only	0x00000000
0x98	New Data 1 Register	MCAN_NDAT1	Read/Write	0x00000000
0x9C	New Data 2 Register	MCAN_NDAT2	Read/Write	0x00000000
0xA0	Receive FIFO 0 Configuration Register	MCAN_RXF0C	Read/Write	0x00000000
0xA4	Receive FIFO 0 Status Register	MCAN_RXF0S	Read-only	0x00000000
0xA8	Receive FIFO 0 Acknowledge Register	MCAN_RXF0A	Read/Write	0x00000000
0xAC	Receive Rx Buffer Configuration Register	MCAN_RXBC	Read/Write	0x00000000

Downloaded from Arrow.com.

Table 47-13. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0xB0	Receive FIFO 1 Configuration Register	MCAN_RXF1C	Read/Write	0x00000000
0xB4	Receive FIFO 1 Status Register	MCAN_RXF1S	Read-only	0x00000000
0xB8	Receive FIFO 1 Acknowledge Register	MCAN_RXF1A	Read/Write	0x00000000
0xBC	Receive Buffer / FIFO Element Size Configuration Register	MCAN_RXESC	Read/Write	0x00000000
0xC0	Transmit Buffer Configuration Register	MCAN_TXBC	Read/Write	0x00000000
0xC4	Transmit FIFO/Queue Status Register	MCAN_TXFQS	Read-only	0x00000000
0xC8	Transmit Buffer Element Size Configuration Register	MCAN_TXESC	Read/Write	0x00000000
0xCC	Transmit Buffer Request Pending Register	MCAN_TXBRP	Read-only	0x00000000
0xD0	Transmit Buffer Add Request Register	MCAN_TXBAR	Read/Write	0x00000000
0xD4	Transmit Buffer Cancellation Request Register	MCAN_TXBCR	Read/Write	0x00000000
0xD8	Transmit Buffer Transmission Occurred Register	MCAN_TXBTO	Read-only	0x00000000
0xDC	Transmit Buffer Cancellation Finished Register	MCAN_TXBCF	Read-only	0x00000000
0xE0	Transmit Buffer Transmission Interrupt Enable Register	MCAN_TXBTIE	Read/Write	0x00000000
0xE4	Transmit Buffer Cancellation Finished Interrupt Enable Register	MCAN_TXBCIE	Read/Write	0x00000000
0xE8-0xEC	Reserved	–	–	–
0xF0	Transmit Event FIFO Configuration Register	MCAN_TXEFC	Read/Write	0x00000000
0xF4	Transmit Event FIFO Status Register	MCAN_TXEFS	Read-only	0x00000000
0xF8	Transmit Event FIFO Acknowledge Register	MCAN_TXEFA	Read/Write	0x00000000
0xFC	Reserved	–	–	–

Notes:
1. Due to clock domain crossing, there is a delay between when a register bit or field is written and when the related status register bits are updated.
2. The reset value for bit 7, MCAN_TEST.RX, is undefined.

Downloaded from Arrow.com.

47.6.1 MCAN Core Release Register

Name: MCAN_CREL

Address: 0x40030000 (0), 0x40034000 (1)

Access: Read-only

31	30	29	28	27	26	25	24
REL				STEP			
23	22	21	20	19	18	17	16
SUBSTEP				YEAR			
15	14	13	12	11	10	9	8
MON							
7	6	5	4	3	2	1	0
DAY							

- **DAY: Timestamp Day**

Two digits, BCD-coded. This field is set by generic parameter on MCAN synthesis.

- **MON: Timestamp Month**

Two digits, BCD-coded. This field is set by generic parameter on MCAN synthesis.

- **YEAR: Timestamp Year**

One digit, BCD-coded. This field is set by generic parameter on MCAN synthesis.

- **SUBSTEP: Sub-step of Core Release**

One digit, BCD-coded.

- **STEP: Step of Core Release**

One digit, BCD-coded.

- **REL: Core Release**

One digit, BCD-coded.

Downloaded from Arrow.com.

47.6.2 MCAN Endian Register

Name: MCAN_ENDN

Address: 0x40030004 (0), 0x40034004 (1)

Access: Read-only

31	30	29	28	27	26	25	24
ETV							
23	22	21	20	19	18	17	16
ETV							
15	14	13	12	11	10	9	8
ETV							
7	6	5	4	3	2	1	0
ETV							

- **ETV: Endianness Test Value**

The endianness test value is 0x87654321.

Downloaded from Arrow.com.

47.6.3 MCAN Customer Register

Name: MCAN_CUST

Address: 0x40030008 (0), 0x40034008 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
CSV							
23	22	21	20	19	18	17	16
CSV							
15	14	13	12	11	10	9	8
CSV							
7	6	5	4	3	2	1	0
CSV							

- **CSV: Customer-specific Value**

Customer-specific value.

Downloaded from Arrow.com.

47.6.4 MCAN Fast Bit Timing and Prescaler Register

Name: MCAN_FBTP

Address: 0x4003000C (0), 0x4003400C (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	TDCO				
23	22	21	20	19	18	17	16
TDC	–	–	FBRP				
15	14	13	12	11	10	9	8
–	–	–	–	FTSEG1			
7	6	5	4	3	2	1	0
–	FTSEG2			–	–	FSJW	

This register can only be written if the bits CCE and INIT are set in MCAN CC Control Register.

The CAN bit time may be programmed in the range of 4 to 25 time quanta. The CAN time quantum may be programmed in the range of 1 to 32 CAN core clock periods. t_q = (FBRP + 1) CAN core clock periods.

FTSEG1 is the sum of Prop_Seg and Phase_Seg1. FTSEG2 is Phase_Seg2.

Therefore the length of the bit time is (programmed values) [FTSEG1 + FTSEG2 + 3] t_q
or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] t_q.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

- **FSJW: Fast (Re) Synchronization Jump Width**

The duration of a synchronization jump is t_q x (FSJW + 1).

- **FTSEG2: Fast Time Segment After Sample Point**

The duration of time segment is t_q x (FTSEG2 + 1).

- **FTSEG1: Fast Time Segment Before Sample Point**

0: Forbidden.

1 to 15: The duration of time segment is t_q x (FTSEG1 + 1).

- **FBRP: Fast Baud Rate Prescaler**

The value by which the peripheral clock is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 31. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

- **TDC: Transceiver Delay Compensation**

0 (DISABLED): Transceiver Delay Compensation disabled.

1 (ENABLED): Transceiver Delay Compensation enabled.

- **TDCO: Transceiver Delay Compensation Offset**

0 to 31: Offset value, in CAN core clock periods, defining the distance between the measured delay from CANTX to CANRX and the secondary sample point.

Notes: 1. With a CAN core clock frequency of 8 MHz, the reset value of 0x00000A33 configures the MCAN for a fast bit rate of 500 kbit/s.

Downloaded from Arrow.com.

2. The bit rate configured for the CAN FD data phase via MCAN_FBTP must be higher than or equal to the bit rate configured for the arbitration phase via MCAN_BTP.

47.6.5 MCAN Test Register

Name: MCAN_TEST

Address: 0x40030010 (0), 0x40034010 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	TDCV					
7	6	5	4	3	2	1	0
RX	TX		LBCK	–	–	–	–

Write access to the Test Register has to be enabled by setting bit MCAN_CCCR.TEST to ‘1’.

All MCAN Test Register functions are set to their reset values when bit MCAN_CCCR.TEST is cleared.

Loop Back mode and software control of pin CANTX are hardware test modes. Programming of TX ≠ 0 disturbs the message transfer on the CAN bus.

- **LBCK: Loop Back Mode (read/write)**

0 (DISABLED): Reset value. Loop Back mode is disabled.

1 (ENABLED): Loop Back mode is enabled (see Section 47.5.1.9).

- **TX: Control of Transmit Pin (read/write)**

Value	Name	Description
0	RESET	Reset value, CANTX controlled by the CAN Core, updated at the end of the CAN bit time.
1	SAMPLE_POINT_MONITORING	Sample Point can be monitored at pin CANTX.
2	DOMINANT	Dominant (‘0’) level at pin CANTX.
3	RECESSIVE	Recessive (‘1’) at pin CANTX.

- **RX: Receive Pin (read-only)**

Monitors the actual value of pin CANRX.

0: The CAN bus is dominant (CANRX = ‘0’).

1: The CAN bus is recessive (CANRX = ‘1’).

- **TDCV: Transceiver Delay Compensation Value (read-only)**

0 to 63: Position of the secondary sample point, in CAN core clock periods, defined by the sum of the measured delay from CANTX to CANRX and MCAN_FBTP.TDCO.

Downloaded from Arrow.com.

47.6.6 MCAN RAM Watchdog Register

Name: MCAN_RWD

Address: 0x40030014 (0), 0x40034014 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
WDV							
7	6	5	4	3	2	1	0
WDC							

The RAM Watchdog monitors the Message RAM response time. A Message RAM access via the MCAN's Generic Master Interface starts the Message RAM Watchdog Counter with the value configured by MCAN_RWD.WDC. The counter is reloaded with MCAN_RWD.WDC when the Message RAM signals successful completion by activating its READY output. In case there is no response from the Message RAM until the counter has counted down to zero, the counter stops and interrupt flag MCAN_IR.WDI is set. The RAM Watchdog Counter is clocked by the system bus clock (peripheral clock).

- **WDC: Watchdog Configuration (read/write)**

Start value of the Message RAM Watchdog Counter. The counter is disabled when WDC is cleared.

- **WDV: Watchdog Value (read-only)**

Watchdog Counter Value for the current message located in RAM.

Downloaded from Arrow.com.

47.6.7 MCAN CC Control Register

Name: MCAN_CCCR

Address: 0x40030018 (0), 0x40034018 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	TXP	FDBS	FDO	CMR		CME	
7	6	5	4	3	2	1	0
TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT

- **INIT: Initialization (read/write)**

0 (DISABLED): Normal operation.

1 (ENABLED): Initialization is started.

Note: Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer has to ensure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

- **CCE: Configuration Change Enable (read/write, write protection)**

0 (PROTECTED): The processor has no write access to the protected configuration registers.

1 (CONFIGURABLE): The processor has write access to the protected configuration registers (while MCAN_CCCR.INIT = ‘1’).

- **ASM: Restricted Operation Mode (read/write, write protection against ‘1’)**

For a description of the Restricted Operation mode see Section 47.5.1.5.

0 (NORMAL): Normal CAN operation.

1 (RESTRICTED): Restricted operation mode active.

- **CSA: Clock Stop Acknowledge (read-only)**

0: No clock stop acknowledged.

1: MCAN may be set in power down by stopping the peripheral clock and the CAN core clock.

- **CSR: Clock Stop Request (read/write)**

0 (NO_CLOCK_STOP): No clock stop is requested.

1 (CLOCK_STOP): Clock stop requested. When clock stop is requested, first INIT and then CSA will be set after all pending transfer requests have been completed and the CAN bus reached idle.

- **MON: Bus Monitoring Mode (read/write, write protection against ‘1’)**

0 (DISABLED): Bus Monitoring mode is disabled.

1 (ENABLED): Bus Monitoring mode is enabled.

Downloaded from Arrow.com.

- **DAR: Disable Automatic Retransmission (read/write, write protection)**

0 (AUTO_RETX): Automatic retransmission of messages not transmitted successfully enabled.

1 (NO_AUTO_RETX): Automatic retransmission disabled.

- **TEST: Test Mode Enable (read/write, write protection against '1')**

0 (DISABLED): Normal operation, MCAN_TEST register holds reset values.

1 (ENABLED): Test mode, write access to MCAN_TEST register enabled.

- **CME: CAN Mode Enable (read/write, write protection)**

Value	Name	Description
0	ISO11898_1	CAN operation according to ISO11898-1 enabled
1	FD	CAN FD operation enabled
2-3	FD_BITRATE_SWITCH	CAN FD operation with bit rate switching enabled

Notes:
1. When CME = 0, received frames are strictly interpreted according to ISO11898-1, which leads to the transmission of an error frame when receiving a CAN FD frame. In case CME = 1, transmission of long CAN FD frames and reception of long and fast CAN FD frames is enabled. With CME = 2 or 3, transmission and reception of long and fast CAN FD frames is enabled.
2. Write protection against '1' indicates that the bit can only be set by the processor when both CCE and INIT are set. The bit can be cleared by the processor at any time.

- **CMR: CAN Mode Request (read/write)**

A change of the CAN operation mode is requested by writing to this field. After change to the requested operation mode, the field is cleared and the status flags FDBS and FDO are set accordingly. In case the requested CAN operation mode is not enabled, the value written to CMR is retained until it is overwritten by the next mode change request. In case CME = 1, 2, 3, a change to CAN operation according to ISO11898-1 is always possible. Default is CAN operation according to ISO11898-1.

Value	Name	Description
0	NO_CHANGE	No mode change
1	FD	Request CAN FD operation
2	FD_BITRATE_SWITCH	Request CAN FD operation with bit rate switching
3	ISO11898_1	Request CAN operation according ISO11898-1

- **FDO: CAN FD Operation (read-only)**

0: This node transmits all frames in CAN format according to ISO11898-1.

1: This node transmits all frames (excl. remote frames) in CAN FD format.

- **FDBS: CAN FD Bit Rate Switching (read-only)**

0: This node transmits no frames with bit rate switching.

1: This node transmits all frames (excl. remote frames) with bit rate switching.

- **TXP: Transmit Pause (read/write, write protection)**

If this bit is set, the MCAN pauses for two CAN bit times before starting the next transmission after itself has successfully transmitted a frame (see Section 47.5.5).

0: Transmit pause disabled.

1: Transmit pause enabled.

Downloaded from Arrow.com.

47.6.8 MCAN Bit Timing and Prescaler Register

Name: MCAN_BTP

Address: 0x4003001C (0), 0x4003401C (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	BRP	
23	22	21	20	19	18	17	16
BRP							
15	14	13	12	11	10	9	8
–	–	TSEG1					
7	6	5	4	3	2	1	0
TSEG2				SJW			

This register can only be written if the bits CCE and INIT are set in MCAN_CCCR.

The CAN bit time may be programmed in the range of 4 to 81 time quanta. The CAN time quantum may be programmed in the range of 1 to 1024 CAN core clock periods. $t_q = t_{core\ clock} \times (BRP + 1)$.

TSEG1 is the sum of Prop_Seg and Phase_Seg1. TSEG2 is Phase_Seg2.

Therefore the length of the bit time is (programmed values) [TSEG1 + TSEG2 + 3] t_q
or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] t_q.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

- **SJW: (Re) Synchronization Jump Width**

0 to 15: The duration of a synchronization jump is $t_q \times (SJW + 1)$.

- **TSEG2: Time Segment After Sample Point**

0 to 15: The duration of time segment is $t_q \times (TSEG2 + 1)$.

- **TSEG1: Time Segment Before Sample Point**

0: Forbidden.

1 to 63: The duration of time segment is $t_q \times (TSEG1 + 1)$.

- **BRP: Baud Rate Prescaler**

0 to 1023: The value by which the oscillator frequency is divided for generating the CAN time quanta. The CAN time is built up from a multiple of this quanta. CAN time quantum (tq) = $t_{core\ clock} \times (BRP + 1)$

Note: With a CAN core clock of 8 MHz, the reset value of 0x00000A33 configures the MCAN for a bit rate of 500 kbit/s.

Downloaded from Arrow.com.

47.6.9 MCAN Timestamp Counter Configuration Register

Name: MCAN_TSCC

Address: 0x40030020 (0), 0x40034020 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	TCP			
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	TSS	

For a description of the Timestamp Counter see Section 47.5.2.

- **TSS: Timestamp Select**

Value	Name	Description
0	ALWAYS_0	Timestamp counter value always 0x0000
1	TCP_INC	Timestamp counter value incremented according to TCP
2	EXT_TIMESTAMP	External timestamp counter value used

- **TCP: Timestamp Counter Prescaler**

Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1…16]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

Note: With CAN FD, an external counter is required for timestamp generation (TSS = 2).

Downloaded from Arrow.com.

47.6.10 MCAN Timestamp Counter Value Register

Name: MCAN_TSCV

Address: 0x40030024 (0), 0x40034024 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
TSC							
7	6	5	4	3	2	1	0
TSC							

- **TSC: Timestamp Counter (cleared on write)**

The internal/external Timestamp Counter value is captured on start of frame (both Receive and Transmit). When MCAN_TSCC.TSS = 1, the Timestamp Counter is incremented in multiples of CAN bit times [1…16] depending on the configuration of MCAN_TSCC.TCP. A wrap around sets interrupt flag MCAN_IR.TSW.

When MCAN_TSCC.TSS = 2, TSC reflects the external Timestamp Counter value. Thus a write access has no impact.

Note: A “wrap around” is a change of the Timestamp Counter value from non-zero to zero not caused by write access to MCAN_TSCV.

Downloaded from Arrow.com.

47.6.11 MCAN Timeout Counter Configuration Register

Name: MCAN_TOCC

Address: 0x40030028 (0), 0x40034028 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
TOP							
23	22	21	20	19	18	17	16
TOP							
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	TOS		ETOC

This register can only be written if the bits CCE and INIT are set in MCAN CC Control Register.

For a description of the Timeout Counter, see Section 47.5.3.

- **ETOC: Enable Timeout Counter**

0 (NO_TIMEOUT): Timeout Counter disabled.

1 (TOS_CONTROLLED): Timeout Counter enabled.

For use of timeout function with CAN FD, see Section 47.5.3.

- **TOS: Timeout Select**

When operating in Continuous mode, a write to MCAN_TOCV presets the counter to the value configured by MCAN_TOCC.TOP and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by MCAN_TOCC.TOP. Down-counting is started when the first FIFO element is stored.

Value	Name	Description
0	CONTINUOUS	Continuous operation
1	TX_EV_TIMEOUT	Timeout controlled by Tx Event FIFO
2	RX0_EV_TIMEOUT	Timeout controlled by Receive FIFO 0
3	RX1_EV_TIMEOUT	Timeout controlled by Receive FIFO 1

- **TOP: Timeout Period**

Start value of the Timeout Counter (down-counter). Configures the Timeout Period.

Downloaded from Arrow.com.

47.6.12 MCAN Timeout Counter Value Register

Name: MCAN_TOCV

Address: 0x4003002C (0), 0x4003402C (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
TOC							
7	6	5	4	3	2	1	0
TOC							

- **TOC: Timeout Counter (cleared on write)**

The Timeout Counter is decremented in multiples of CAN bit times [1…16] depending on the configuration of MCAN_TSCC.TCP. When decremented to zero, interrupt flag MCAN_IR.TOO is set and the Timeout Counter is stopped. Start and reset/restart conditions are configured via MCAN_TOCC.TOS.

Downloaded from Arrow.com.

47.6.13 MCAN Error Counter Register

Name: MCAN_ECR

Address: 0x40030040 (0), 0x40034040 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
CEL							
15	14	13	12	11	10	9	8
RP	REC						
7	6	5	4	3	2	1	0
TEC							

- **TEC: Transmit Error Counter**

Actual state of the Transmit Error Counter, values between 0 and 255.

- **REC: Receive Error Counter**

Actual state of the Receive Error Counter, values between 0 and 127.

- **RP: Receive Error Passive**

0: The Receive Error Counter is below the error passive level of 128.

1: The Receive Error Counter has reached the error passive level of 128.

- **CEL: CAN Error Logging (cleared on read)**

The counter is incremented each time when a CAN protocol error causes the Transmit Error Counter or the Receive Error Counter to be incremented. It is reset by read access to CEL. The counter stops at 0xFF; the next increment of TEC or REC sets interrupt flag IR.ELO.

Note: When MCAN_CCCR.ASM is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented.

Downloaded from Arrow.com.

47.6.14 MCAN Protocol Status Register

Name: MCAN_PSR

Address: 0x40030044 (0), 0x40034044 (1)

Access: Read-only/

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	REDL	RBRS	RESI	FLEC		
7	6	5	4	3	2	1	0
BO	EW	EP	ACT		LEC		

- **LEC: Last Error Code (set to 111 on read)**

The LEC indicates the type of the last error to occur on the CAN bus. This field is cleared when a message has been transferred (reception or transmission) without error.

Value	Name	Description
0	NO_ERROR	No error occurred since LEC has been reset by successful reception or transmission.
1	STUFF_ERROR	More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.
2	FORM_ERROR	A fixed format part of a received frame has the wrong format.
3	ACK_ERROR	The message transmitted by the MCAN was not acknowledged by another node.
4	BIT1_ERROR	During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value ‘1’), but the monitored bus value was dominant.
5	BIT0_ERROR	During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value ‘0’), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the processor to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).
6	CRC_ERROR	The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.
7	NO_CHANGE	Any read access to the Protocol Status Register re-initializes the LEC to ‘7’. When the LEC shows the value ‘7’, no CAN bus event was detected since the last processor read access to the Protocol Status Register.

- **ACT: Activity**

Monitors the CAN communication state of the CAN module.

Value	Name	Description
0	SYNCHRONIZING	Node is synchronizing on CAN communication
1	IDLE	Node is neither receiver nor transmitter
2	RECEIVER	Node is operating as receiver
3	TRANSMITTER	Node is operating as transmitter

Downloaded from Arrow.com.

- **EP: Error Passive**

0: The MCAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected.

1: The MCAN is in the Error_Passive state.

- **EW: Warning Status**

0: Both error counters are below the Error_Warning limit of 96.

1: At least one of error counter has reached the Error_Warning limit of 96.

- **BO: Bus_Off Status**

0: The MCAN is not Bus_Off.

1: The MCAN is in Bus_Off state.

- **FLEC: Fast Last Error Code (set to 111 on read)**

Type of last error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC. This field will be cleared to zero when a CAN FD format frame with its BRS flag set has been transferred (reception or transmission) without error.

- **RESI: ESI Flag of Last Received CAN FD Message (cleared on read)**

This bit is set together with REDL, independently from acceptance filtering.

0: Last received CAN FD message did not have its ESI flag set.

1: Last received CAN FD message had its ESI flag set.

- **RBRS: BRS Flag of Last Received CAN FD Message (cleared on read)**

This bit is set together with REDL, independently from acceptance filtering.

0: Last received CAN FD message did not have its BRS flag set.

1: Last received CAN FD message had its BRS flag set.

- **REDL: Received a CAN FD Message (cleared on read)**

This bit is set independently from acceptance filtering.

0: Since this bit was reset by the processor, no CAN FD message has been received.

1: Message in CAN FD format with EDL flag set has been received.

Notes:
1. When a frame in CAN FD format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) will be shown in FLEC instead of LEC. An error in a fixed stuff bit of a CAN FD CRC sequence will be shown as a Form Error, not Stuff Error.
2. The Bus_Off recovery sequence (see *CAN Specification Rev. 2.0 or ISO11898-1*) cannot be shortened by setting or resetting MCAN_CCCR.INIT. If the device goes Bus_Off, it will set MCAN_CCCR.INIT of its own accord, stopping all bus activities. Once MCAN_CCCR.INIT has been cleared by the processor, the device will then wait for 129 occurrences of Bus Idle (129 * 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of MCAN_CCCR.INIT, each time a sequence of 11 recessive bits has been monitored, a Bit0 Error code is written to MCAN_PSR.LEC, enabling the processor to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. MCAN_ECR.REC is used to count these sequences.

Downloaded from Arrow.com.

47.6.15 MCAN Interrupt Register

Name: MCAN_IR

Address: 0x40030050 (0), 0x40034050 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
STE	FOE	ACKE	BE	CRCE	WDI	BO	EW
23	22	21	20	19	18	17	16
EP	ELO	–	–	DRX	TOO	MRAF	TSW
15	14	13	12	11	10	9	8
TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM
7	6	5	4	3	2	1	0
RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the processor clears them. A flag is cleared by writing a ‘1’ to the corresponding bit position. Writing a ‘0’ has no effect. A hard reset will clear the register. The configuration of IE controls whether an interrupt is generated. The configuration of ILS controls on which interrupt line an interrupt is signalled.

- **RF0N: Receive FIFO 0 New Message**

0: No new message written to Receive FIFO 0.

1: New message written to Receive FIFO 0.

- **RF0W: Receive FIFO 0 Watermark Reached**

0: Receive FIFO 0 fill level below watermark.

1: Receive FIFO 0 fill level reached watermark.

- **RF0F: Receive FIFO 0 Full**

0: Receive FIFO 0 not full.

1: Receive FIFO 0 full.

- **RF0L: Receive FIFO 0 Message Lost**

0: No Receive FIFO 0 message lost.

1: Receive FIFO 0 message lost, also set after write attempt to Receive FIFO 0 of size zero.

- **RF1N: Receive FIFO 1 New Message**

0: No new message written to Receive FIFO 1.

1: New message written to Receive FIFO 1.

- **RF1W: Receive FIFO 1 Watermark Reached**

0: Receive FIFO 1 fill level below watermark.

1: Receive FIFO 1 fill level reached watermark.

- **RF1F: Receive FIFO 1 Full**

0: Receive FIFO 1 not full.

1: Receive FIFO 1 full.

Downloaded from Arrow.com.

- **RF1L: Receive FIFO 1 Message Lost**

0: No Receive FIFO 1 message lost.

1: Receive FIFO 1 message lost, also set after write attempt to Receive FIFO 1 of size zero.

- **HPM: High Priority Message**

0: No high priority message received.

1: High priority message received.

- **TC: Transmission Completed**

0: No transmission completed.

1: Transmission completed.

- **TCF: Transmission Cancellation Finished**

0: No transmission cancellation finished.

1: Transmission cancellation finished.

- **TFE: Tx FIFO Empty**

0: Tx FIFO non-empty.

1: Tx FIFO empty.

- **TEFN: Tx Event FIFO New Entry**

0: Tx Event FIFO unchanged.

1: Tx Handler wrote Tx Event FIFO element.

- **TEFW: Tx Event FIFO Watermark Reached**

0: Tx Event FIFO fill level below watermark.

1: Tx Event FIFO fill level reached watermark.

- **TEFF: Tx Event FIFO Full**

0: Tx Event FIFO not full.

1: Tx Event FIFO full.

- **TEFL: Tx Event FIFO Element Lost**

0: No Tx Event FIFO element lost.

1: Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero.

- **TSW: Timestamp Wraparound**

0: No timestamp counter wrap-around.

1: Timestamp counter wrapped around.

Downloaded from Arrow.com.

- **MRAF: Message RAM Access Failure**

The flag is set, when the Rx Handler

- has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx Handler starts processing of the following message.
- was not able to write a message to the Message RAM. In this case message storage is aborted.

In both cases the FIFO put index is not updated resp. the New Data flag for a dedicated Receive Buffer is not set, a partly stored message is overwritten when the next message is stored to this location.

The flag is also set when the Tx Handler was not able to read a message from the Message RAM in time. In this case message transmission is aborted. In case of a Tx Handler access failure the MCAN is switched into Restricted Operation mode (see Section 47.5.1.5). To leave Restricted Operation mode, the processor has to reset MCAN_CCCR.ASM.

0: No Message RAM access failure occurred.

1: Message RAM access failure occurred.

- **TOO: Timeout Occurred**

0: No timeout.

1: Timeout reached.

- **DRX: Message stored to Dedicated Receive Buffer**

The flag is set whenever a received message has been stored into a dedicated Receive Buffer.

0: No Receive Buffer updated.

1: At least one received message stored into a Receive Buffer.

- **ELO: Error Logging Overflow**

0: CAN Error Logging Counter did not overflow.

1: Overflow of CAN Error Logging Counter occurred.

- **EP: Error Passive**

0: Error_Passive status unchanged.

1: Error_Passive status changed.

- **EW: Warning Status**

0: Error_Warning status unchanged.

1: Error_Warning status changed.

- **BO: Bus_Off Status**

0: Bus_Off status unchanged.

1: Bus_Off status changed.

- **WDI: Watchdog Interrupt**

0: No Message RAM Watchdog event occurred.

1: Message RAM Watchdog event due to missing READY.

Downloaded from Arrow.com.

- **CRCE: CRC Error**

0: No CRC Error detected.

1: Received CRC did not match the calculated CRC.

- **BE: Bit Error**

0: No Bit Error detected.

1: Device wanted to send a rec / dom level, but monitored bus level was dominant / recessive.

- **ACKE: Acknowledge Error**

0: No Acknowledge Error detected.

1: A transmitted message was not acknowledged by another node.

- **FOE: Format Error**

0: No Format Error detected.

1: A fixed format part of a received frame has the wrong format.

- **STE: Stuff Error**

0: No Stuff Error detected.

1: More than 5 equal bits in a sequence occurred.

Downloaded from Arrow.com.

47.6.16 MCAN Interrupt Enable Register

Name: MCAN_IE

Address: 0x40030054 (0), 0x40034054 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
STEE	FOEE	ACKEE	BEE	CRCEE	WDIE	BOE	EWE
23	22	21	20	19	18	17	16
EPE	ELOE	–	–	DRXE	TOOE	MRAFE	TSWE
15	14	13	12	11	10	9	8
TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME
7	6	5	4	3	2	1	0
RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE

The following configuration values are valid for all listed bit names of this register:

0: Disables the corresponding interrupt.

1: Enables the corresponding interrupt.

- **RF0NE: Receive FIFO 0 New Message Interrupt Enable**
- **RF0WE: Receive FIFO 0 Watermark Reached Interrupt Enable**
- **RF0FE: Receive FIFO 0 Full Interrupt Enable**
- **RF0LE: Receive FIFO 0 Message Lost Interrupt Enable**
- **RF1NE: Receive FIFO 1 New Message Interrupt Enable**
- **RF1WE: Receive FIFO 1 Watermark Reached Interrupt Enable**
- **RF1FE: Receive FIFO 1 Full Interrupt Enable**
- **RF1LE: Receive FIFO 1 Message Lost Interrupt Enable**
- **HPME: High Priority Message Interrupt Enable**
- **TCE: Transmission Completed Interrupt Enable**
- **TCFE: Transmission Cancellation Finished Interrupt Enable**
- **TFEE: Tx FIFO Empty Interrupt Enable**
- **TEFNE: Tx Event FIFO New Entry Interrupt Enable**
- **TEFWE: Tx Event FIFO Watermark Reached Interrupt Enable**
- **TEFFE: Tx Event FIFO Full Interrupt Enable**
- **TEFLE: Tx Event FIFO Event Lost Interrupt Enable**
- **TSWE: Timestamp Wraparound Interrupt Enable**

Downloaded from Arrow.com.

- **MRAFE: Message RAM Access Failure Interrupt Enable**
- **TOOE: Timeout Occurred Interrupt Enable**
- **DRXE: Message stored to Dedicated Receive Buffer Interrupt Enable**
- **ELOE: Error Logging Overflow Interrupt Enable**
- **EPE: Error Passive Interrupt Enable**
- **EWE: Warning Status Interrupt Enable**
- **BOE: Bus_Off Status Interrupt Enable**
- **WDIE: Watchdog Interrupt Enable**
- **CRCEE: CRC Error Interrupt Enable**
- **BEE: Bit Error Interrupt Enable**
- **ACKEE: Acknowledge Error Interrupt Enable**
- **FOEE: Format Error Interrupt Enable**
- **STEE: Stuff Error Interrupt Enable**

Downloaded from Arrow.com.

47.6.17 MCAN Interrupt Line Select Register

Name: MCAN_ILS

Address: 0x40030058 (0), 0x40034058 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
STEL	FOEL	ACKEL	BEL	CRCEL	WDIL	BOL	EWL
23	22	21	20	19	18	17	16
EPL	ELOL	–	–	DRXL	TOOL	MRAFL	TSWL
15	14	13	12	11	10	9	8
TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML
7	6	5	4	3	2	1	0
RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL

The Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the Interrupt Register to one of the two module interrupt lines.

0: Interrupt assigned to interrupt line m_can_int0.

1: Interrupt assigned to interrupt line m_can_int1.

- **RF0NL: Receive FIFO 0 New Message Interrupt Line**
- **RF0WL: Receive FIFO 0 Watermark Reached Interrupt Line**
- **RF0FL: Receive FIFO 0 Full Interrupt Line**
- **RF0LL: Receive FIFO 0 Message Lost Interrupt Line**
- **RF1NL: Receive FIFO 1 New Message Interrupt Line**
- **RF1WL: Receive FIFO 1 Watermark Reached Interrupt Line**
- **RF1FL: Receive FIFO 1 Full Interrupt Line**
- **RF1LL: Receive FIFO 1 Message Lost Interrupt Line**
- **HPML: High Priority Message Interrupt Line**
- **TCL: Transmission Completed Interrupt Line**
- **TCFL: Transmission Cancellation Finished Interrupt Line**
- **TFEL: Tx FIFO Empty Interrupt Line**
- **TEFNL: Tx Event FIFO New Entry Interrupt Line**
- **TEFWL: Tx Event FIFO Watermark Reached Interrupt Line**
- **TEFFL: Tx Event FIFO Full Interrupt Line**
- **TEFLL: Tx Event FIFO Event Lost Interrupt Line**
- **TSWL: Timestamp Wraparound Interrupt Line**

Downloaded from Arrow.com.

- **MRAFL: Message RAM Access Failure Interrupt Line**
- **TOOL: Timeout Occurred Interrupt Line**
- **DRXL: Message stored to Dedicated Receive Buffer Interrupt Line**
- **ELOL: Error Logging Overflow Interrupt Line**
- **EPL: Error Passive Interrupt Line**
- **EWL: Warning Status Interrupt Line**
- **BOL: Bus_Off Status Interrupt Line**
- **WDIL: Watchdog Interrupt Line**
- **CRCEL: CRC Error Interrupt Line**
- **BEL: Bit Error Interrupt Line**
- **ACKEL: Acknowledge Error Interrupt Line**
- **FOEL: Format Error Interrupt Line**
- **STEL: Stuff Error Interrupt Line**

Downloaded from Arrow.com.

47.6.18 MCAN Interrupt Line Enable

Name: MCAN_ILE

Address: 0x4003005C (0), 0x4003405C (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	EINT1	EINT0

Each of the two interrupt lines to the processor can be enabled / disabled separately by programming bits EINT0 and EINT1.

- **EINT0: Enable Interrupt Line 0**

0: Interrupt line m_can_int0 disabled.

1: Interrupt line m_can_int0 enabled.

- **EINT1: Enable Interrupt Line 1**

0: Interrupt line m_can_int1 disabled.

1: Interrupt line m_can_int1 enabled.

Downloaded from Arrow.com.

47.6.19 MCAN Global Filter Configuration

Name: MCAN_GFC

Address: 0x40030080 (0), 0x40034080 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	ANFS		ANFE		RRFS	RRFE

This register can only be written if the bits CCE and INIT are set in MCAN CC Control Register.

- **RRFE: Reject Remote Frames Extended**

0 (FILTER): Filter remote frames with 29-bit extended IDs.

1 (REJECT): Reject all remote frames with 29-bit extended IDs.

- **RRFS: Reject Remote Frames Standard**

0 (FILTER): Filter remote frames with 11-bit standard IDs.

1 (REJECT): Reject all remote frames with 11-bit standard IDs.

- **ANFE: Accept Non-matching Frames Extended**

Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated.

Value	Name	Description
0	RX_FIFO_0	Message stored in Receive FIFO 0
1	RX_FIFO_1	Message stored in Receive FIFO 1
2-3	REJECTED	Message rejected

- **ANFS: Accept Non-matching Frames Standard**

Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated.

Value	Name	Description
0	RX_FIFO_0	Message stored in Receive FIFO 0
1	RX_FIFO_1	Message stored in Receive FIFO 1
2-3	REJECTED	Message rejected

Downloaded from Arrow.com.

47.6.20 MCAN Standard ID Filter Configuration

Name: MCAN_SIDFC

Address: 0x40030084 (0), 0x40034084 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
LSS							
15	14	13	12	11	10	9	8
FLSSA							
7	6	5	4	3	2	1	0
FLSSA						–	–

This register can only be written if the bits CCE and INIT are set in MCAN CC Control Register.

- **FLSSA: Filter List Standard Start Address**

Start address of standard Message ID filter list (32-bit word address, see Figure 47-12).

Write FLSSA with the bits [15:2] of the 32-bit address.

- **LSS: List Size Standard**

0: No standard Message ID filter.

1-128: Number of standard Message ID filter elements.

>128: Values greater than 128 are interpreted as 128.

Downloaded from Arrow.com.

47.6.21 MCAN Extended ID Filter Configuration

Name: MCAN_XIDFC

Address: 0x40030088 (0), 0x40034088 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	LSE						
15	14	13	12	11	10	9	8
FLESA							
7	6	5	4	3	2	1	0
FLESA						–	–

This register can only be written if the bits CCE and INIT are set in MCAN CC Control Register.

- **FLESA: Filter List Extended Start Address**

Start address of extended Message ID filter list (32-bit word address, see Figure 47-12).

Write FLESA with the bits [15:2] of the 32-bit address.

- **LSE: List Size Extended**

0: No extended Message ID filter.

1-64: Number of extended Message ID filter elements.

>64: Values greater than 64 are interpreted as 64.

Downloaded from Arrow.com.

47.6.22 MCAN Extended ID AND Mask

Name: MCAN_XIDAM

Address: 0x40030090 (0), 0x40034090 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	EIDM				
23	22	21	20	19	18	17	16
EIDM							
15	14	13	12	11	10	9	8
EIDM							
7	6	5	4	3	2	1	0
EIDM							

This register can only be written if the bits CCE and INIT are set in MCAN CC Control Register.

- **EIDM: Extended ID Mask**

For acceptance filtering of extended frames the Extended ID AND Mask is ANDed with the Message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to one the mask is not active.

Downloaded from Arrow.com.

47.6.23 MCAN High Priority Message Status

Name: MCAN_HPMS

Address: 0x40030094 (0), 0x40034094 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
FLST	FIDX						
7	6	5	4	3	2	1	0
MSI	BIDX						

This register is updated every time a Message ID filter element configured to generate a priority event matches. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

- **BIDX: Buffer Index**

Index of Receive FIFO element to which the message was stored. Only valid when MSI[1] = ‘1’.

- **MSI: Message Storage Indicator**

Value	Name	Description
0	NO_FIFO_SEL	No FIFO selected.
1	LOST	FIFO message.
2	FIFO_0	Message stored in FIFO 0.
3	FIFO_1	Message stored in FIFO 1.

- **FIDX: Filter Index**

Index of matching filter element. Range is 0 to MCAN_SIDFC.LSS - 1 resp. MCAN_XIDFC.LSE - 1.

- **FLST: Filter List**

Indicates the filter list of the matching filter element.

0: Standard filter list

1: Extended filter list

Downloaded from Arrow.com.

47.6.24 MCAN New Data 1

Name: MCAN_NDAT1

Address: 0x40030098 (0), 0x40034098 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24
23	22	21	20	19	18	17	16
ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16
15	14	13	12	11	10	9	8
ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8
7	6	5	4	3	2	1	0
ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0

- **NDx: New Data**

The register holds the New Data flags of Receive Buffers 0 to 31. The flags are set when the respective Receive Buffer has been updated from a received frame. The flags remain set until the processor clears them. A flag is cleared by writing a ‘1’ to the corresponding bit position. Writing a ‘0’ has no effect. A hard reset will clear the register.

0: Receive Buffer not updated

1: Receive Buffer updated from new message

Downloaded from Arrow.com.

47.6.25 MCAN New Data 2

Name: MCAN_NDAT2

Address: 0x4003009C (0), 0x4003409C (1)

Access: Read/Write

31	30	29	28	27	26	25	24
ND63	ND62	ND61	ND60	ND59	ND58	ND57	ND56
23	22	21	20	19	18	17	16
ND55	ND54	ND53	ND52	ND51	ND50	ND49	ND48
15	14	13	12	11	10	9	8
ND47	ND46	ND45	ND44	ND43	ND42	ND41	ND40
7	6	5	4	3	2	1	0
ND39	ND38	ND37	ND36	ND35	ND34	ND33	ND32

- **NDx: New Data**

The register holds the New Data flags of Receive Buffers 32 to 63. The flags are set when the respective Receive Buffer has been updated from a received frame. The flags remain set until the processor clears them. A flag is cleared by writing a ‘1’ to the corresponding bit position. Writing a ‘0’ has no effect. A hard reset will clear the register.

0: Receive Buffer not updated.

1: Receive Buffer updated from new message.

Downloaded from Arrow.com.

47.6.26 MCAN Receive FIFO 0 Configuration

Name: MCAN_RXF0C

Address: 0x400300A0 (0), 0x400340A0 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
F0OM	F0WM						
23	22	21	20	19	18	17	16
–	F0S						
15	14	13	12	11	10	9	8
F0SA							
7	6	5	4	3	2	1	0
F0SA						–	–

This register can only be written if the bits CCE and INIT are set in MCAN CC Control Register.

- **F0SA: Receive FIFO 0 Start Address**

Start address of Receive FIFO 0 in Message RAM (32-bit word address, see Figure 47-12).

Write F0SA with the bits [15:2] of the 32-bit address.

- **F0S: Receive FIFO 0 Size**

0: No Receive FIFO 0

1-64: Number of Receive FIFO 0 elements.

>64: Values greater than 64 are interpreted as 64.

The Receive FIFO 0 elements are indexed from 0 to F0S-1.

- **F0WM: Receive FIFO 0 Watermark**

0: Watermark interrupt disabled.

1-64: Level for Receive FIFO 0 watermark interrupt (MCAN_IR.RF0W).

>64: Watermark interrupt disabled.

- **F0OM: FIFO 0 Operation Mode**

FIFO 0 can be operated in blocking or in overwrite mode (see Section 47.5.4.2).

0: FIFO 0 blocking mode.

1: FIFO 0 overwrite mode.

Downloaded from Arrow.com.

47.6.27 MCAN Receive FIFO 0 Status

Name: MCAN_RXF0S

Address: 0x400300A4 (0), 0x400340A4 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	RF0L	F0F
23	22	21	20	19	18	17	16
–	–	F0PI					
15	14	13	12	11	10	9	8
–	–	F0GI					
7	6	5	4	3	2	1	0
–	F0FL						

- **F0FL: Receive FIFO 0 Fill Level**

Number of elements stored in Receive FIFO 0, range 0 to 64.

- **F0GI: Receive FIFO 0 Get Index**

Receive FIFO 0 read index pointer, range 0 to 63.

- **F0PI: Receive FIFO 0 Put Index**

Receive FIFO 0 write index pointer, range 0 to 63.

- **F0F: Receive FIFO 0 Full**

0: Receive FIFO 0 not full.

1: Receive FIFO 0 full.

- **RF0L: Receive FIFO 0 Message Lost**

This bit is a copy of interrupt flag MCAN_IR.RF0L. When MCAN_IR.RF0L is reset, this bit is also reset.

0: No Receive FIFO 0 message lost

1: Receive FIFO 0 message lost, also set after write attempt to Receive FIFO 0 of size zero

Note: Overwriting the oldest message when MCAN_RXF0C.F0OM = ‘1’ will not set this flag.

Downloaded from Arrow.com.

47.6.28 MCAN Receive FIFO 0 Acknowledge

Name: MCAN_RXF0A

Address: 0x400300A8 (0), 0x400340A8 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	F0AI					

- **F0AI: Receive FIFO 0 Acknowledge Index**

After the processor has read a message or a sequence of messages from Receive FIFO 0 it has to write the buffer index of the last element read from Receive FIFO 0 to F0AI. This will set the Receive FIFO 0 Get Index MCAN_RXF0S.F0GI to F0AI + 1 and update the FIFO 0 Fill Level MCAN_RXF0S.F0FL.

Downloaded from Arrow.com.

47.6.29 MCAN Receive Buffer Configuration

Name: MCAN_RXBC

Address: 0x400300AC (0), 0x400340AC (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
RBSA							
7	6	5	4	3	2	1	0
RBSA						–	–

- **RBSA: Receive Buffer Start Address**

Configures the start address of the Receive Buffers section in the Message RAM (32-bit word address, see Figure 47-12). Also used to reference debug messages A,B,C.

Write RBSA with the bits [15:2] of the 32-bit address.

Downloaded from Arrow.com.

47.6.30 MCAN Receive FIFO 1 Configuration

Name: MCAN_RXF1C

Address: 0x400300B0 (0), 0x400340B0 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
F1OM	F1WM						
23	22	21	20	19	18	17	16
–	F1S						
15	14	13	12	11	10	9	8
F1SA							
7	6	5	4	3	2	1	0
F1SA						–	–

This register can only be written if the bits CCE and INIT are set in MCAN CC Control Register.

- **F1SA: Receive FIFO 1 Start Address**

Start address of Receive FIFO 1 in Message RAM (32-bit word address, see Figure 47-12).

Write F1SA with the bits [15:2] of the 32-bit address.

- **F1S: Receive FIFO 1 Size**

0: No Receive FIFO 1

1-64: Number of elements in Receive FIFO 1.

>64: Values greater than 64 are interpreted as 64.

The elements in Receive FIFO 1 are indexed from 0 to F1S - 1.

- **F1WM: Receive FIFO 1 Watermark**

0: Watermark interrupt disabled

1-64: Level for Receive FIFO 1 watermark interrupt (MCAN_IR.RF1W).

>64: Watermark interrupt disabled.

- **F1OM: FIFO 1 Operation Mode**

FIFO 1 can be operated in blocking or in overwrite mode (see Section 47.5.4.2).

0: FIFO 1 blocking mode.

1: FIFO 1 overwrite mode.

Downloaded from Arrow.com.

47.6.31 MCAN Receive FIFO 1 Status

Name: MCAN_RXF1S

Address: 0x400300B4 (0), 0x400340B4 (1)

Access: Read-only

31	30	29	28	27	26	25	24
DMS		–	–	–	–	RF1L	F1F
23	22	21	20	19	18	17	16
–	–	F1PI					
15	14	13	12	11	10	9	8
–	–	F1GI					
7	6	5	4	3	2	1	0
–	F1FL						

- **F1FL: Receive FIFO 1 Fill Level**

Number of elements stored in Receive FIFO 1, range 0 to 64.

- **F1GI: Receive FIFO 1 Get Index**

Receive FIFO 1 read index pointer, range 0 to 63.

- **F1PI: Receive FIFO 1 Put Index**

Receive FIFO 1 write index pointer, range 0 to 63.

- **F1F: Receive FIFO 1 Full**

0: Receive FIFO 1 not full.

1: Receive FIFO 1 full.

- **RF1L: Receive FIFO 1 Message Lost**

This bit is a copy of interrupt flag IR.RF1L. When IR.RF1L is reset, this bit is also reset.

0: No Receive FIFO 1 message lost.

1: Receive FIFO 1 message lost, also set after write attempt to Receive FIFO 1 of size zero.

Note: Overwriting the oldest message when MCAN_RXF1C.F1OM = ‘1’ will not set this flag.

- **DMS: Debug Message Status**

Value	Name	Description
0	IDLE	Idle state, wait for reception of debug messages, DMA request is cleared.
1	MSG_A	Debug message A received.
2	MSG_AB	Debug messages A, B received.
3	MSG_ABC	Debug messages A, B, C received, DMA request is set.

Downloaded from Arrow.com.

47.6.32 MCAN Receive FIFO 1 Acknowledge

Name: MCAN_RXF1A

Address: 0x400300B8 (0), 0x400340B8 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	F1AI					

- **F1AI: Receive FIFO 1 Acknowledge Index**

After the processor has read a message or a sequence of messages from Receive FIFO 1 it has to write the buffer index of the last element read from Receive FIFO 1 to F1AI. This will set the Receive FIFO 1 Get Index MCAN_RXF1S.F1GI to F1AI + 1 and update the FIFO 1 Fill Level MCAN_RXF1S.F1FL.

Downloaded from Arrow.com.

47.6.33 MCAN Receive Buffer / FIFO Element Size Configuration

Name: MCAN_RXESC

Address: 0x400300BC (0), 0x400340BC (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	RBDS		
7	6	5	4	3	2	1	0
–	F1DS			–	F0DS		

This register can only be written if the bits CCE and INIT are set in MCAN CC Control Register.

Configures the number of data bytes belonging to a Receive Buffer / Receive FIFO element. Data field sizes >8 bytes are intended for CAN FD operation only.

- **F0DS: Receive FIFO 0 Data Field Size**

Value	Name	Description
0	8_BYTE	8-byte data field
1	12_BYTE	12-byte data field
2	16_BYTE	16-byte data field
3	20_BYTE	20-byte data field
4	24_BYTE	24-byte data field
5	32_BYTE	32-byte data field
6	48_BYTE	48-byte data field
7	64_BYTE	64-byte data field

- **F1DS: Receive FIFO 1 Data Field Size**

Value	Name	Description
0	8_BYTE	8-byte data field
1	12_BYTE	12-byte data field
2	16_BYTE	16-byte data field
3	20_BYTE	20-byte data field
4	24_BYTE	24-byte data field
5	32_BYTE	32-byte data field
6	48_BYTE	48-byte data field
7	64_BYTE	64-byte data field

Downloaded from Arrow.com.

- **RBDS: Receive Buffer Data Field Size**

Value	Name	Description
0	8_BYTE	8-byte data field
1	12_BYTE	12-byte data field
2	16_BYTE	16-byte data field
3	20_BYTE	20-byte data field
4	24_BYTE	24-byte data field
5	32_BYTE	32-byte data field
6	48_BYTE	48-byte data field
7	64_BYTE	64-byte data field

Note: In case the data field size of an accepted CAN frame exceeds the data field size configured for the matching Receive Buffer or Receive FIFO, only the number of bytes as configured by MCAN_RXESC are stored to the Receive Buffer resp. Receive FIFO element. The rest of the frame's data field is ignored.

Downloaded from Arrow.com.

47.6.34 MCAN Tx Buffer Configuration

Name: MCAN_TXBC

Address: 0x400300C0 (0), 0x400340C0 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	TFQM	TFQS					
23	22	21	20	19	18	17	16
–	–	NDTB					
15	14	13	12	11	10	9	8
TBSA							
7	6	5	4	3	2	1	0
TBSA						–	–

This register can only be written if the bits CCE and INIT are set in MCAN CC Control Register.

- **TBSA: Tx Buffers Start Address**

Start address of Tx Buffers section in Message RAM (32-bit word address, see Figure 47-12).

Write TBSA with the bits [15:2] of the 32-bit address.

- **NDTB: Number of Dedicated Transmit Buffers**

0: No dedicated Tx Buffers.

1-32: Number of dedicated Tx Buffers.

>32: Values greater than 32 are interpreted as 32.

- **TFQS: Transmit FIFO/Queue Size**

0: No Tx FIFO/Queue.

1-32: Number of Tx Buffers used for Tx FIFO/Queue.

>32: Values greater than 32 are interpreted as 32.

- **TFQM: Tx FIFO/Queue Mode**

0: Tx FIFO operation.

1: Tx Queue operation.

Note: Be aware that the sum of TFQS and NDTB may be not greater than 32. There is no check for erroneous configurations. The Tx Buffers section in the Message RAM starts with the dedicated Tx Buffers.

Downloaded from Arrow.com.

47.6.35 MCAN Tx FIFO/Queue Status

Name: MCAN_TXFQS

Address: 0x400300C4 (0), 0x400340C4 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	TFQF	TFQPI				
15	14	13	12	11	10	9	8
–	–	–	TFGI				
7	6	5	4	3	2	1	0
–	–	TFFL					

The Tx FIFO/Queue status is related to the pending Tx requests listed in register MCAN_TXBRP. Therefore the effect of Add/Cancellation requests may be delayed due to a running Tx scan (MCAN_TXBRP not yet updated).

- **TFFL: Tx FIFO Free Level**

Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 32. Read as zero when Tx Queue operation is configured (MCAN_TXBC.TFQM = ‘1’).

- **TFGI: Tx FIFO Get Index**

Tx FIFO read index pointer, range 0 to 31. Read as zero when Tx Queue operation is configured (MCAN_TXBC.TFQM = ‘1’).

- **TFQPI: Tx FIFO/Queue Put Index**

Tx FIFO/Queue write index pointer, range 0 to 31.

- **TFQF: Tx FIFO/Queue Full**

0: Tx FIFO/Queue not full.

1: Tx FIFO/Queue full.

Note: In case of mixed configurations where dedicated Tx Buffers are combined with a Tx FIFO or a Tx Queue, the Put and Get Indices indicate the number of the Tx Buffer starting with the first dedicated Tx Buffers.
Example: For a configuration of 12 dedicated Tx Buffers and a Tx FIFO of 20 Buffers a Put Index of 15 points to the fourth buffer of the Tx FIFO.

Downloaded from Arrow.com.

47.6.36 MCAN Tx Buffer Element Size Configuration

Name: MCAN_TXESC

Address: 0x400300C8 (0), 0x400340C8 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	TBDS		

This register can only be written if the bits CCE and INIT are set in MCAN CC Control Register.

Configures the number of data bytes belonging to a Tx Buffer element. Data field sizes > 8 bytes are intended for CAN FD operation only.

- **TBDS: Tx Buffer Data Field Size**

Value	Name	Description
0	8_BYTE	8-byte data field
1	12_BYTE	12-byte data field
2	16_BYTE	16-byte data field
3	20_BYTE	20-byte data field
4	24_BYTE	24-byte data field
5	32_BYTE	32-byte data field
6	48_BYTE	48- byte data field
7	64_BYTE	64-byte data field

Note: In case the data length code DLC of a Tx Buffer element is configured to a value higher than the Tx Buffer data field size MCAN_TXESC.TBDS, the bytes not defined by the Tx Buffer are transmitted as “0xCC” (padding bytes).

Downloaded from Arrow.com.

47.6.37 MCAN Transmit Buffer Request Pending

Name: MCAN_TXBRP

Address: 0x400300CC (0), 0x400340CC (1)

Access: Read-only

31	30	29	28	27	26	25	24
TRP31	TRP30	TRP29	TRP28	TRP27	TRP26	TRP25	TRP24
23	22	21	20	19	18	17	16
TRP23	TRP22	TRP21	TRP20	TRP19	TRP18	TRP17	TRP16
15	14	13	12	11	10	9	8
TRP15	TRP14	TRP13	TRP12	TRP11	TRP10	TRP9	TRP8
7	6	5	4	3	2	1	0
TRP7	TRP6	TRP5	TRP4	TRP3	TRP2	TRP1	TRP0

- **TRPx: Transmission Request Pending for Buffer x**

Each Tx Buffer has its own Transmission Request Pending bit. The bits are set via register MCAN_TXBAR. The bits are reset after a requested transmission has completed or has been cancelled via register MCAN_TXBCR.

TXBRP bits are set only for those Tx Buffers configured via MCAN_TXBC. After a MCAN_TXBRP bit has been set, a Tx scan (see Section 47.5.5) is started to check for the pending Tx request with the highest priority (Tx Buffer with lowest Message ID).

A cancellation request resets the corresponding transmission request pending bit of register MCAN_TXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding TXBRP bit has been reset.

After a cancellation has been requested, a finished cancellation is signalled via MCAN_TXBCF.

- after successful transmission together with the corresponding MCAN_TXBTO bit.
- when the transmission has not yet been started at the point of cancellation.
- when the transmission has been aborted due to lost arbitration.
- when an error occurred during frame transmission.

In DAR mode all transmissions are automatically cancelled if they are not successful. The corresponding MCAN_TXBCF bit is set for all unsuccessful transmissions.

0: No transmission request pending

1: Transmission request pending

Note: MCAN_TXBRP bits which are set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx Buffer, this Add Request is cancelled immediately, the corresponding MCAN_TXBRP bit is reset.

Downloaded from Arrow.com.

47.6.38 MCAN Transmit Buffer Add Request

Name: MCAN_TXBAR

Address: 0x400300D0 (0), 0x400340D0 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
AR31	AR30	AR29	AR28	AR27	AR26	AR25	AR24
23	22	21	20	19	18	17	16
AR23	AR22	AR21	AR20	AR19	AR18	AR17	AR16
15	14	13	12	11	10	9	8
AR15	AR14	AR13	AR12	AR11	AR10	AR9	AR8
7	6	5	4	3	2	1	0
AR7	AR6	AR5	AR4	AR3	AR2	AR1	AR0

- **ARx: Add Request for Transmit Buffer x**

Each Transmit Buffer has its own Add Request bit. Writing a ‘1’ will set the corresponding Add Request bit; writing a ‘0’ has no impact. This enables the processor to set transmission requests for multiple Transmit Buffers with one write to MCAN_TXBAR. MCAN_TXBAR bits are set only for those Transmit Buffers configured via TXBC. When no Transmit scan is running, the bits are reset immediately, else the bits remain set until the Transmit scan process has completed.

0: No transmission request added.

1: Transmission requested added.

Note: If an add request is applied for a Transmit Buffer with pending transmission request (corresponding MCAN_TXBRP bit already set), this Add Request is ignored.

Downloaded from Arrow.com.

47.6.39 MCAN Transmit Buffer Cancellation Request

Name: MCAN_TXBCR

Address: 0x400300D4 (0), 0x400340D4 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
CR31	CR30	CR29	CR28	CR27	CR26	CR25	CR24
23	22	21	20	19	18	17	16
CR23	CR22	CR21	CR20	CR19	CR18	CR17	CR16
15	14	13	12	11	10	9	8
CR15	CR14	CR13	CR12	CR11	CR10	CR9	CR8
7	6	5	4	3	2	1	0
CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0

- **CRx: Cancellation Request for Transmit Buffer x**

Each Transmit Buffer has its own Cancellation Request bit. Writing a ‘1’ will set the corresponding Cancellation Request bit; writing a ‘0’ has no impact. This enables the processor to set cancellation requests for multiple Transmit Buffers with one write to MCAN_TXBCR. MCAN_TXBCR bits are set only for those Transmit Buffers configured via TXBC. The bits remain set until the corresponding bit of MCAN_TXBRP is reset.

0: No cancellation pending.

1: Cancellation pending.

Downloaded from Arrow.com.

47.6.40 MCAN Transmit Buffer Transmission Occurred

Name: MCAN_TXBTO

Address: 0x400300D8 (0), 0x400340D8 (1)

Access: Read-only

31	30	29	28	27	26	25	24
TO31	TO30	TO29	TO28	TO27	TO26	TO25	TO24
23	22	21	20	19	18	17	16
TO23	TO22	TO21	TO20	TO19	TO18	TO17	TO16
15	14	13	12	11	10	9	8
TO15	TO14	TO13	TO12	TO11	TO10	TO9	TO8
7	6	5	4	3	2	1	0
TO7	TO6	TO5	TO4	TO3	TO2	TO1	TO0

- **TOx: Transmission Occurred for Buffer x**

Each Transmit Buffer has its own Transmission Occurred bit. The bits are set when the corresponding MCAN_TXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a ‘1’ to the corresponding bit of register MCAN_TXBAR.

0: No transmission occurred.

1: Transmission occurred.

Downloaded from Arrow.com.

47.6.41 MCAN Transmit Buffer Cancellation Finished

Name: MCAN_TXBCF

Address: 0x400300DC (0), 0x400340DC (1)

Access: Read-only

31	30	29	28	27	26	25	24
CF31	CF30	CF29	CF28	CF27	CF26	CF25	CF24
23	22	21	20	19	18	17	16
CF23	CF22	CF21	CF20	CF19	CF18	CF17	CF16
15	14	13	12	11	10	9	8
CF15	CF14	CF13	CF12	CF11	CF10	CF9	CF8
7	6	5	4	3	2	1	0
CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0

- **CFx: Cancellation Finished for Transmit Buffer x**

Each Transmit Buffer has its own Cancellation Finished bit. The bits are set when the corresponding MCAN_TXBRP bit is cleared after a cancellation was requested via MCAN_TXBCR. In case the corresponding MCAN_TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing a ‘1’ to the corresponding bit of register MCAN_TXBAR.

0: No transmit buffer cancellation.

1: Transmit buffer cancellation finished.

Downloaded from Arrow.com.

47.6.42 MCAN Transmit Buffer Transmission Interrupt Enable

Name: MCAN_TXBTIE

Address: 0x400300E0 (0), 0x400340E0 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
TIE31	TIE30	TIE29	TIE28	TIE27	TIE26	TIE25	TIE24
23	22	21	20	19	18	17	16
TIE23	TIE22	TIE21	TIE20	TIE19	TIE18	TIE17	TIE16
15	14	13	12	11	10	9	8
TIE15	TIE14	TIE13	TIE12	TIE11	TIE10	TIE9	TIE8
7	6	5	4	3	2	1	0
TIE7	TIE6	TIE5	TIE4	TIE3	TIE2	TIE1	TIE0

- **TIEx: Transmission Interrupt Enable for Buffer x**

Each Transmit Buffer has its own Transmission Interrupt Enable bit.

0: Transmission interrupt disabled

1: Transmission interrupt enable

Downloaded from Arrow.com.

47.6.43 MCAN Transmit Buffer Cancellation Finished Interrupt Enable

Name: MCAN_TXBCIE

Address: 0x400300E4 (0), 0x400340E4 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
CFIE31	CFIE30	CFIE29	CFIE28	CFIE27	CFIE26	CFIE25	CFIE24
23	22	21	20	19	18	17	16
CFIE23	CFIE22	CFIE21	CFIE20	CFIE19	CFIE18	CFIE17	CFIE16
15	14	13	12	11	10	9	8
CFIE15	CFIE14	CFIE13	CFIE12	CFIE11	CFIE10	CFIE9	CFIE8
7	6	5	4	3	2	1	0
CFIE7	CFIE6	CFIE5	CFIE4	CFIE3	CFIE2	CFIE1	CFIE0

- **CFIEx: Cancellation Finished Interrupt Enable for Transmit Buffer x**

Each Transmit Buffer has its own Cancellation Finished Interrupt Enable bit.

0: Cancellation finished interrupt disabled.

1: Cancellation finished interrupt enabled.

Downloaded from Arrow.com.

47.6.44 MCAN Transmit Event FIFO Configuration

Name: MCAN_TXEFC

Address: 0x400300F0 (0), 0x400340F0 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	EFWM					
23	22	21	20	19	18	17	16
–	–	EFS					
15	14	13	12	11	10	9	8
EFSA							
7	6	5	4	3	2	1	0
EFSA						–	–

This register can only be written if the bits CCE and INIT are set in MCAN CC Control Register.

- **EFSA: Event FIFO Start Address**

Start address of Tx Event FIFO in Message RAM (32-bit word address, see Figure 47-12).

Write EFSA with the bits [15:2] of the 32-bit address.

- **EFS: Event FIFO Size**

0: Tx Event FIFO disabled.

1-32: Number of Tx Event FIFO elements.

>32: Values greater than 32 are interpreted as 32.

The Tx Event FIFO elements are indexed from 0 to EFS - 1.

- **EFWM: Event FIFO Watermark**

0: Watermark interrupt disabled.

1-32: Level for Tx Event FIFO watermark interrupt (MCAN_IR.TEFW).

>32: Watermark interrupt disabled.

Downloaded from Arrow.com.

47.6.45 MCAN Tx Event FIFO Status

Name: MCAN_TXEFS

Address: 0x400300F4 (0), 0x400340F4 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	TEFL	EFF
23	22	21	20	19	18	17	16
–	–	–	EFPI				
15	14	13	12	11	10	9	8
–	–	–	EFGI				
7	6	5	4	3	2	1	0
–	–	EFFL					

- **EFFL: Event FIFO Fill Level**

Number of elements stored in Tx Event FIFO, range 0 to 32.

- **EFGI: Event FIFO Get Index**

Tx Event FIFO read index pointer, range 0 to 31.

- **EFPI: Event FIFO Put Index**

Tx Event FIFO write index pointer, range 0 to 31.

- **EFF: Event FIFO Full**

0: Tx Event FIFO not full

1: Tx Event FIFO full

- **TEFL: Tx Event FIFO Element Lost**

This bit is a copy of interrupt flag MCAN_IR.TEFL. When MCAN_IR.TEFL is reset, this bit is also reset.

0: No Tx Event FIFO element lost

1: Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero.

Downloaded from Arrow.com.

47.6.46 MCAN Tx Event FIFO Acknowledge

Name: MCAN_TXEFA

Address: 0x400300F8 (0), 0x400340F8 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	EFAI				

- **EFAI: Event FIFO Acknowledge Index**

After the processor has read an element or a sequence of elements from the Tx Event FIFO, it has to write the index of the last element read from Tx Event FIFO to EFAI. This will set the Tx Event FIFO Get Index MCAN_TXEFS.EFGI to EFAI + 1 and update the FIFO 0 Fill Level MCAN_TXEFS.EFFL.

Downloaded from Arrow.com.

48. Timer Counter (TC)

48.1 Description

A Timer Counter (TC) module includes three identical TC channels. The number of implemented TC modules is device-specific.

Each TC channel can be independently programmed to perform a wide range of functions including frequency measurement, event counting, interval measurement, pulse generation, delay timing and pulse width modulation.

Each channel has three external clock inputs, five internal clock inputs and two multi-purpose input/output signals which can be configured by the user. Each channel drives an internal interrupt signal which can be programmed to generate processor interrupts.

The TC embeds a quadrature decoder (QDEC) connected in front of the timers and driven by TIOA0, TIOB0 and TIOB1 inputs. When enabled, the QDEC performs the input lines filtering, decoding of quadrature signals and connects to the timers/counters in order to read the position and speed of the motor through the user interface.

The TC block has two global registers which act upon all TC channels:

- Block Control Register (TC_BCR)—allows channels to be started simultaneously with the same instruction
- Block Mode Register (TC_BMR)—defines the external clock inputs for each channel, allowing them to be chained

48.2 Embedded Characteristics

- Total number of TC channels implemented on this device: twelve
- TC channel size: 16-bit
- Wide range of functions including:
  - Frequency measurement
  - Event counting
  - Interval measurement
  - Pulse generation
  - Delay timing
  - Pulse Width Modulation
  - Up/down capabilities
  - Quadrature decoder
  - 2-bit Gray up/down count for stepper motor
- Each channel is user-configurable and contains:
  - Three external clock inputs
  - Five Internal clock inputs
  - Two multi-purpose input/output signals acting as trigger event
  - Trigger/capture events can be directly synchronized by PWM signals
- Internal interrupt signal
- Read of the Capture registers by the DMAC
- Compare event fault generation for PWM
- Register Write Protection

Downloaded from Arrow.com.

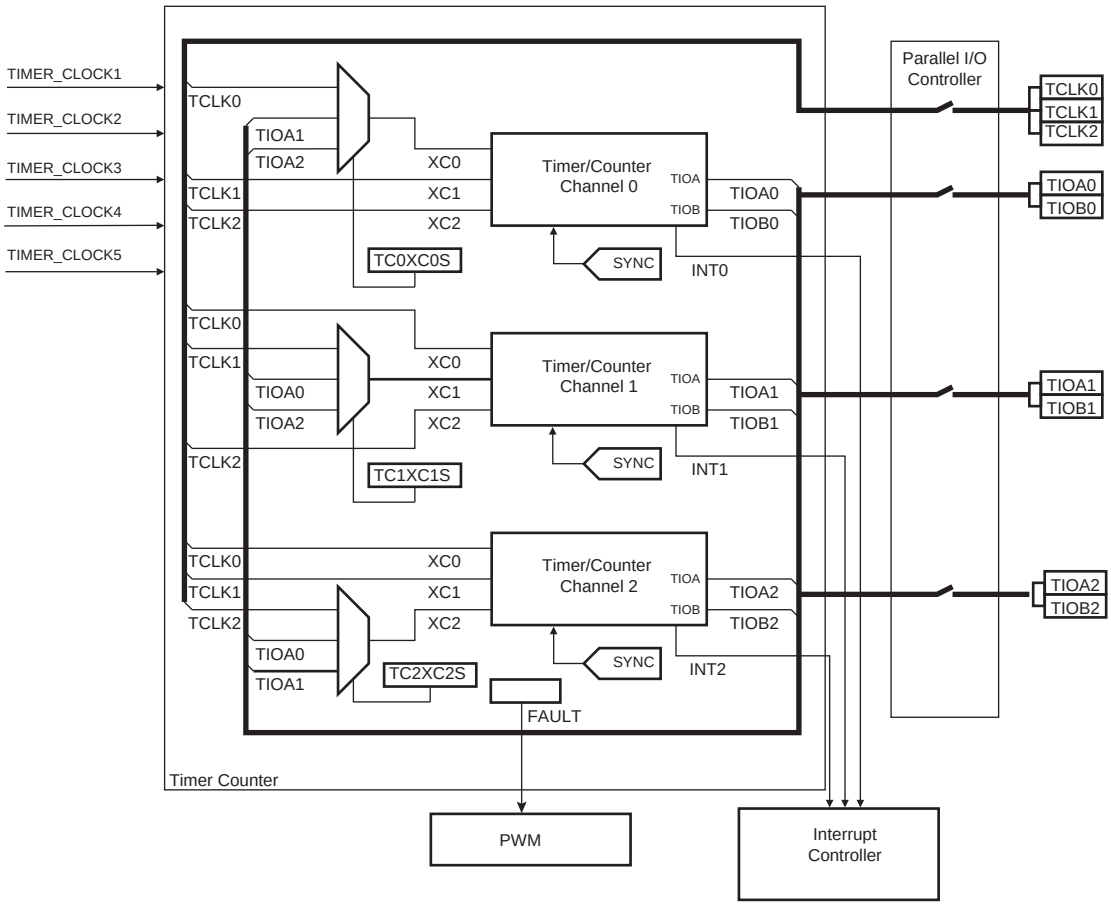
48.3 Block Diagram

Table 48-1. Timer Counter Clock Assignment

Name	Definition
TIMER_CLOCK1	PCK6
TIMER_CLOCK2	MCK/8
TIMER_CLOCK3	MCK/32
TIMER_CLOCK4	MCK/128
TIMER_CLOCK5[(1)]	SLCK

Note: 1. When SLCK is selected for Peripheral Clock (CSS = 0 in PMC Master Clock Register), SLCK input is equivalent to Peripheral Clock.

Figure 48-1. Timer Counter Block Diagram



Note: The QDEC connections are detailed in Figure 48-17.

Downloaded from Arrow.com.

Table 48-2. Signal Description

Block/Channel	Signal Name	Description
Channel Signal	XC0, XC1, XC2	External Clock Inputs
	TIOA	Capture Mode: Timer Counter Input Waveform Mode: Timer Counter Output
	TIOB	Capture Mode: Timer Counter Input Waveform Mode: Timer Counter Input/Output
	INT	Interrupt Signal Output (internal signal)
	SYNC	Synchronization Input Signal (from configuration register)

48.4 Pin List

Table 48-3. Pin List

Pin Name	Description	Type
TCLK0–TCLK2	External Clock Input	Input
TIOA0–TIOA2	I/O Line A	I/O
TIOB0–TIOB2	I/O Line B	I/O

Downloaded from Arrow.com.

48.5 Product Dependencies

48.5.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the TC pins to their peripheral functions.

Table 48-4. I/O Lines

Instance	Signal	I/O Line	Peripheral
TC0	TCLK0	PA4	B
TC0	TCLK1	PA28	B
TC0	TCLK2	PA29	B
TC0	TIOA0	PA0	B
TC0	TIOA1	PA15	B
TC0	TIOA2	PA26	B
TC0	TIOB0	PA1	B
TC0	TIOB1	PA16	B
TC0	TIOB2	PA27	B
TC1	TCLK3	PC25	B
TC1	TCLK4	PC28	B
TC1	TCLK5	PC31	B
TC1	TIOA3	PC23	B
TC1	TIOA4	PC26	B
TC1	TIOA5	PC29	B
TC1	TIOB3	PC24	B
TC1	TIOB4	PC27	B
TC1	TIOB5	PC30	B
TC2	TCLK6	PC7	B
TC2	TCLK7	PC10	B
TC2	TCLK8	PC14	B
TC2	TIOA6	PC5	B
TC2	TIOA7	PC8	B
TC2	TIOA8	PC11	B
TC2	TIOB6	PC6	B
TC2	TIOB7	PC9	B
TC2	TIOB8	PC12	B
TC3	TCLK9	PE2	B
TC3	TCLK10	PE5	B
TC3	TCLK11	PD24	C
TC3	TIOA9	PE0	B
TC3	TIOA10	PE3	B

Downloaded from Arrow.com.

Table 48-4. I/O Lines (Continued)

TC3	TIOA11	PD21	C
TC3	TIOB9	PE1	B
TC3	TIOB10	PE4	B
TC3	TIOB11	PD22	C

48.5.2 Power Management

The TC is clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the Timer Counter clock of each channel.

48.5.3 Interrupt Sources

The TC has an interrupt line per channel connected to the interrupt controller. Handling the TC interrupt requires programming the interrupt controller before configuring the TC.

Table 48-5. Peripheral IDs

Instance	ID
TC0	23
TC1	24
TC2	25
TC3	26

48.5.4 Synchronization Inputs from PWM

The TC has trigger/capture inputs internally connected to the PWM. Refer to Section 48.6.14 “Synchronization with PWM” and to the implementation of the Pulse Width Modulation (PWM) in this product.

48.5.5 Fault Output

The TC has the FAULT output internally connected to the fault input of PWM. Refer to Section 48.6.18 “Fault Mode” and to the implementation of the Pulse Width Modulation (PWM) in this product.

Downloaded from Arrow.com.

48.6 Functional Description

48.6.1 Description

All channels of the Timer Counter are independent and identical in operation except when the QDEC is enabled. The registers for channel programming are listed in Table 48-6 “Register Mapping”.

48.6.2 16-bit Counter

Each 16-bit channel is organized around a 16-bit counter. The value of the counter is incremented at each positive edge of the selected clock. When the counter has reached the value 2^{16}-1 and passes to zero, an overflow occurs and the COVFS bit in the TC Status Register (TC_SR) is set.

The current value of the counter is accessible in real time by reading the TC Counter Value Register (TC_CV). The counter can be reset by a trigger. In this case, the counter value passes to zero on the next valid edge of the selected clock.

48.6.3 Clock Selection

At block level, input clock signals of each channel can either be connected to the external inputs TCLK0, TCLK1 or TCLK2, or be connected to the internal I/O signals TIOA0, TIOA1 or TIOA2 for chaining by programming the TC Block Mode Register (TC_BMR). See Figure 48-2.

Each channel can independently select an internal or external clock source for its counter:

- External clock signals(1): XC0, XC1 or XC2
- Internal clock signals(2): PCK6, MCK/8, MCK/32, MCK/128, SLCK

This selection is made by the TCCLKS bits in the TC Channel Mode Register (TC_CMR).

The selected clock can be inverted with the CLKI bit in the TC_CMR. This allows counting on the opposite edges of the clock.

The burst function allows the clock to be validated when an external signal is high. The BURST parameter in the TC_CMR defines this signal (none, XC0, XC1, XC2). See Figure 48-3.

Note:
1. In all cases, if an external clock is used, the duration of each of its levels must be longer than the peripheral clock period. The external clock frequency must be at least 2.5 times lower than the peripheral clock.
2. In all cases, if asynchronous internal clock PCK6 is used, the duration of each of its levels must be longer than the peripheral clock period. The external clock frequency must be at least 2.5 times lower than the peripheral clock.

Downloaded from Arrow.com.

Figure 48-2. Clock Chaining Selection

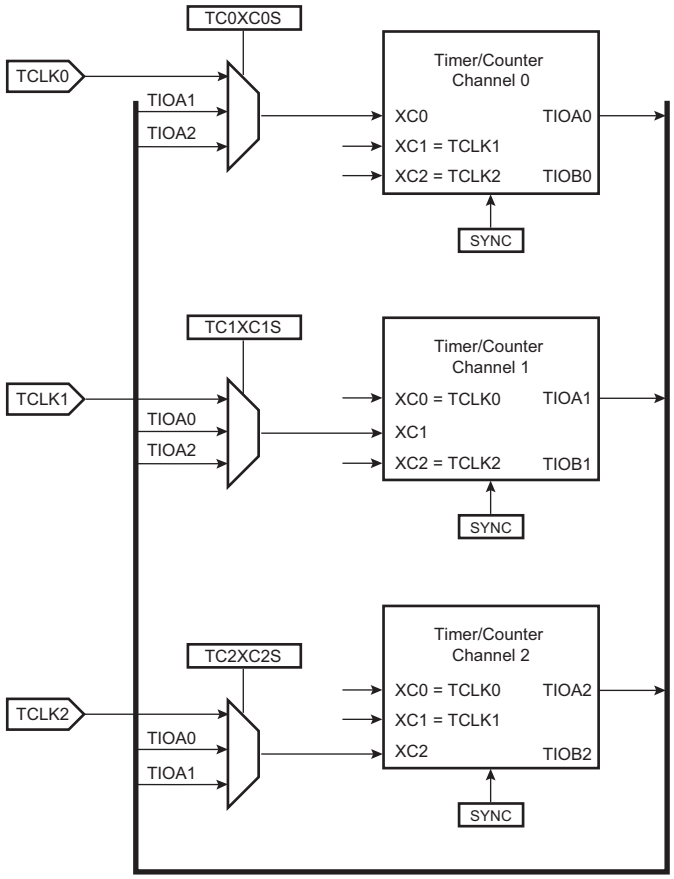
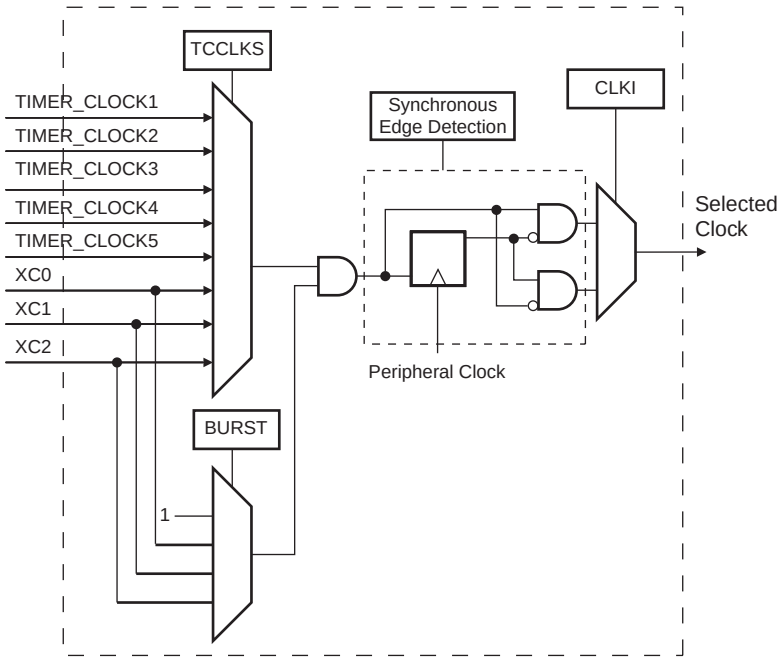


Figure 48-3. Clock Selection



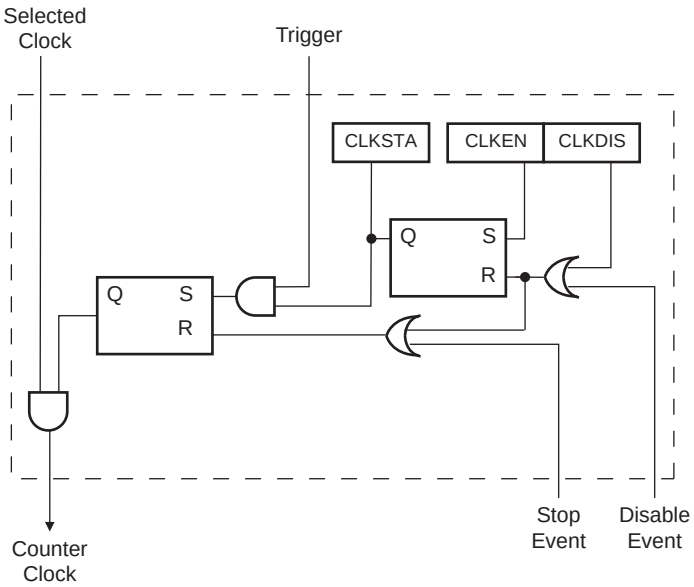
Downloaded from Arrow.com.

48.6.4 Clock Control

The clock of each counter can be controlled in two different ways: it can be enabled/disabled and started/stopped. See Figure 48-4.

- The clock can be enabled or disabled by the user with the CLKEN and the CLKDIS commands in the TC Channel Control Register (TC_CCR). In Capture mode it can be disabled by an RB load event if LDBDIS is set to 1 in the TC_CMR. In Waveform mode, it can be disabled by an RC Compare event if CPCDIS is set to 1 in TC_CMR. When disabled, the start or the stop actions have no effect: only a CLKEN command in the TC_CCR can re-enable the clock. When the clock is enabled, the CLKSTA bit is set in the TC_SR.
- The clock can also be started or stopped: a trigger (software, synchro, external or compare) always starts the clock. The clock can be stopped by an RB load event in Capture mode (LDBSTOP = 1 in TC_CMR) or an RC compare event in Waveform mode (CPCSTOP = 1 in TC_CMR). The start and the stop commands are effective only if the clock is enabled.

Figure 48-4. Clock Control



48.6.5 Operating Modes

Each channel can operate independently in two different modes:

- Capture mode provides measurement on signals.
- Waveform mode provides wave generation.

The TC operating mode is programmed with the WAVE bit in the TC_CMR.

In Capture mode, TIOA and TIOB are configured as inputs.

In Waveform mode, TIOA is always configured to be an output and TIOB is an output if it is not selected to be the external trigger.

48.6.6 Trigger

A trigger resets the counter and starts the counter clock. Three types of triggers are common to both modes, and a fourth external trigger is available to each mode.

Regardless of the trigger used, it will be taken into account at the following active edge of the selected clock. This means that the counter value can be read differently from zero just after a trigger, especially when a low frequency signal is selected as the clock.

Downloaded from Arrow.com.

The following triggers are common to both modes:

- Software Trigger: Each channel has a software trigger, available by setting SWTRG in TC_CCR.
- SYNC: Each channel has a synchronization signal SYNC. When asserted, this signal has the same effect as a software trigger. The SYNC signals of all channels are asserted simultaneously by writing TC_BCR (Block Control) with SYNC set.
- Compare RC Trigger: RC is implemented in each channel and can provide a trigger when the counter value matches the RC value if CPCTRG is set in the TC_CMR.

The channel can also be configured to have an external trigger. In Capture mode, the external trigger signal can be selected between TIOA and TIOB. In Waveform mode, an external event can be programmed on one of the following signals: TIOB, XC0, XC1 or XC2. This external event can then be programmed to perform a trigger by setting bit ENETRG in the TC_CMR.

If an external trigger is used, the duration of the pulses must be longer than the peripheral clock period in order to be detected.

48.6.7 Capture Mode

Capture mode is entered by clearing the WAVE bit in the TC_CMR.

Capture mode allows the TC channel to perform measurements such as pulse timing, frequency, period, duty cycle and phase on TIOA and TIOB signals which are considered as inputs.

Figure 48-6 shows the configuration of the TC channel when programmed in Capture mode.

48.6.8 Capture Registers A and B

Registers A and B (RA and RB) are used as capture registers. They can be loaded with the counter value when a programmable event occurs on the signal TIOA.

The LDRA field in the TC_CMR defines the TIOA selected edge for the loading of register A, and the LDRB field defines the TIOA selected edge for the loading of Register B.

The subsampling ratio defined by the SBSMPLR field in TC_CMR is applied to these selected edges, so that the loading of Register A and Register B occurs once every 1, 2, 4, 8 or 16 selected edges.

RA is loaded only if it has not been loaded since the last trigger or if RB has been loaded since the last loading of RA.

RB is loaded only if RA has been loaded since the last trigger or the last loading of RB.

Loading RA or RB before the read of the last value loaded sets the Overrun Error Flag (LOVRS bit) in the TC_SR. In this case, the old value is overwritten.

When DMA is used, the RAB register address must be configured as source address of the transfer. The RAB register provides the next unread value from Register A and Register B. It may be read by the DMA after a request has been triggered upon loading Register A or Register B.

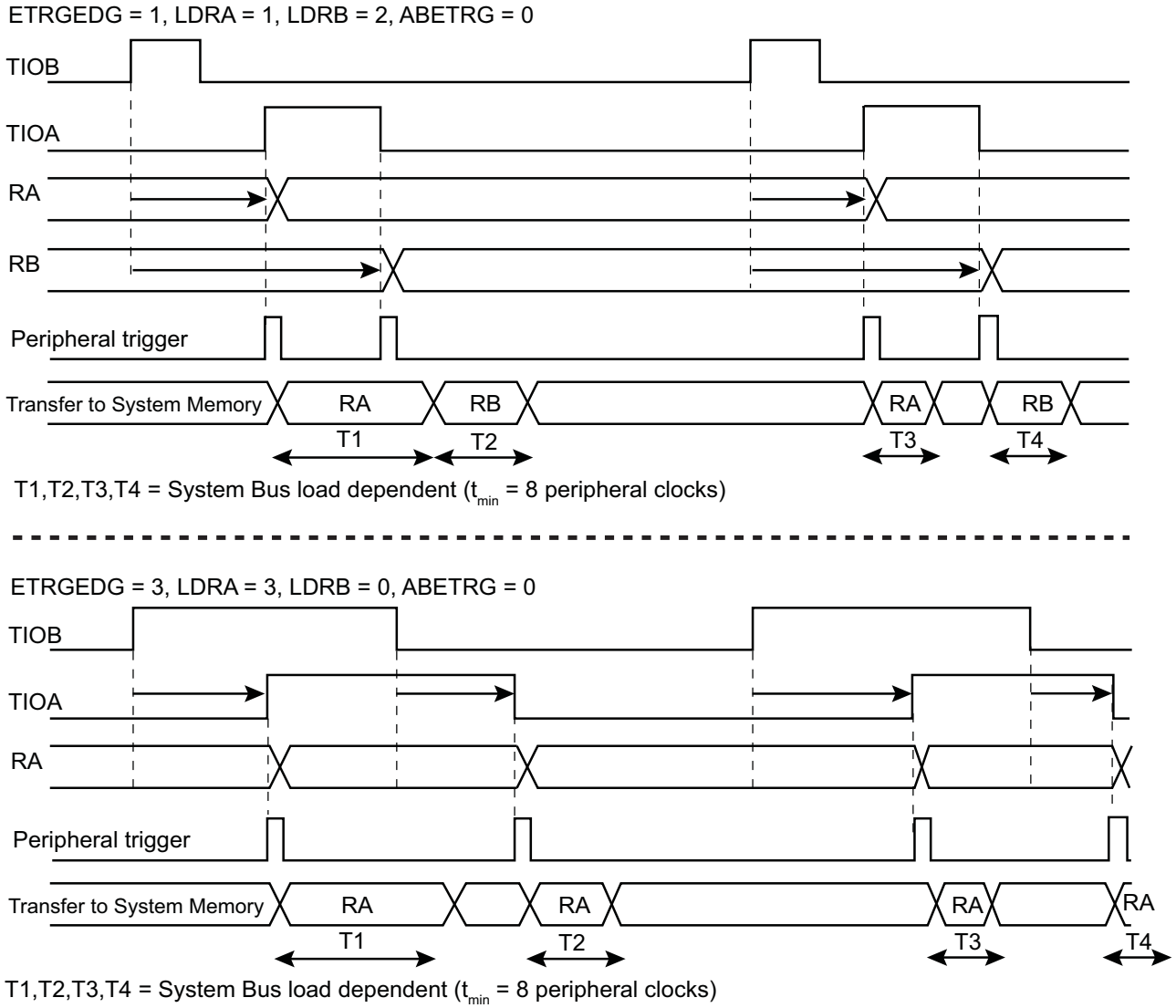
48.6.9 Transfer with DMAC

The DMAC can only perform access from timer to system memory.

Figure 48-5 illustrates how TC_RA and TC_RB can be loaded in the system memory without CPU intervention.

Downloaded from Arrow.com.

Figure 48-5. Example of Transfer with DMAC



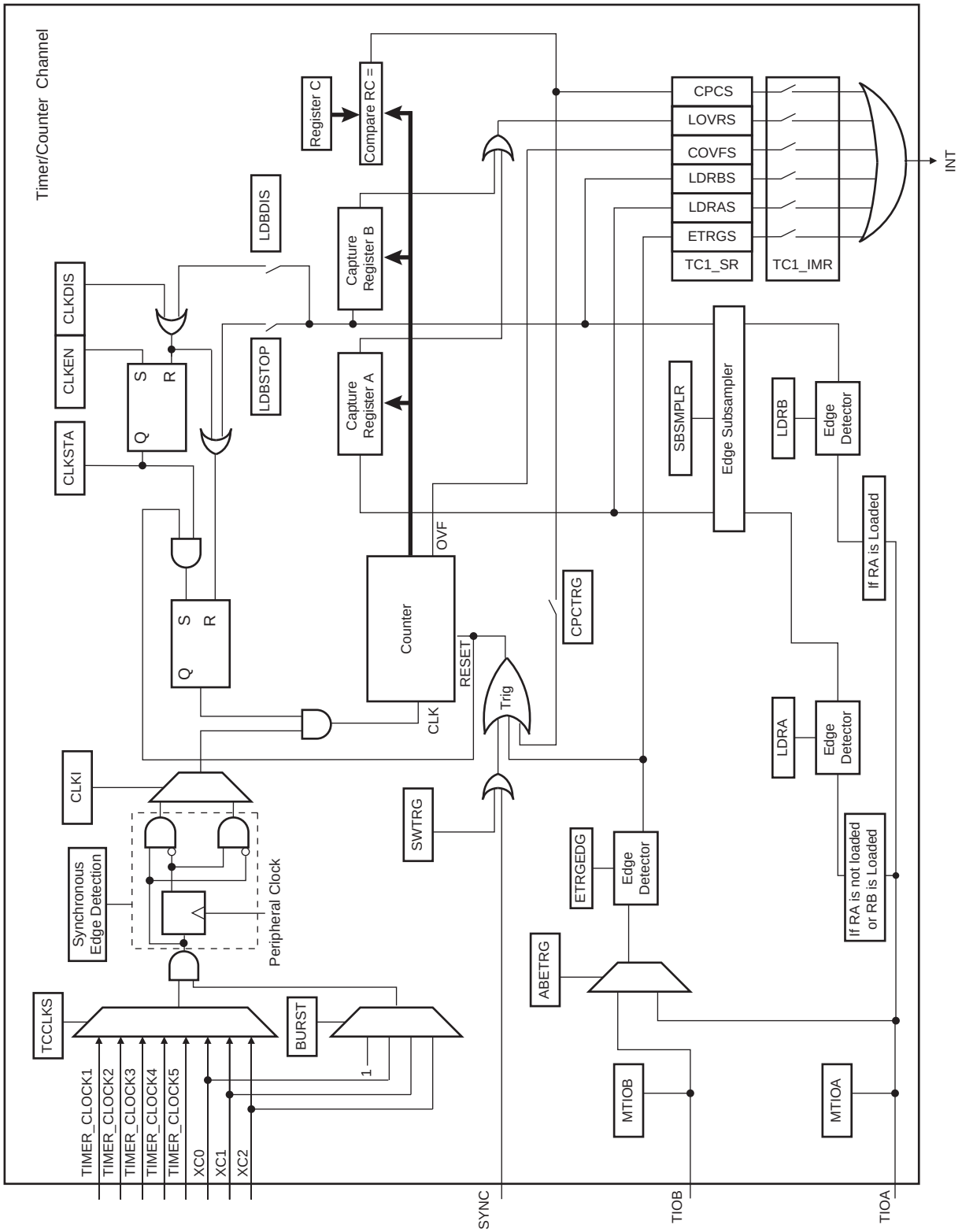
48.6.10 Trigger Conditions

In addition to the SYNC signal, the software trigger and the RC compare trigger, an external trigger can be defined.

The ABETRG bit in the TC_CMR selects TIOA or TIOB input signal as an external trigger or the trigger signal from the output comparator of the PWM module. The External Trigger Edge Selection parameter (ETRGEDG field in TC_CMR) defines the edge (rising, falling, or both) detected to generate an external trigger. If ETRGEDG = 0 (none), the external trigger is disabled.

Downloaded from Arrow.com.

Figure 48-6. Capture Mode



Downloaded from Arrow.com.

48.6.11 Waveform Mode

Waveform mode is entered by setting the TC_CMRx.WAVE bit.

In Waveform mode, the TC channel generates one or two PWM signals with the same frequency and independently programmable duty cycles, or generates different types of one-shot or repetitive pulses.

In this mode, TIOA is configured as an output and TIOB is defined as an output if it is not used as an external event (EEVT parameter in TC_CMR).

Figure 48-7 shows the configuration of the TC channel when programmed in Waveform operating mode.

48.6.12 Waveform Selection

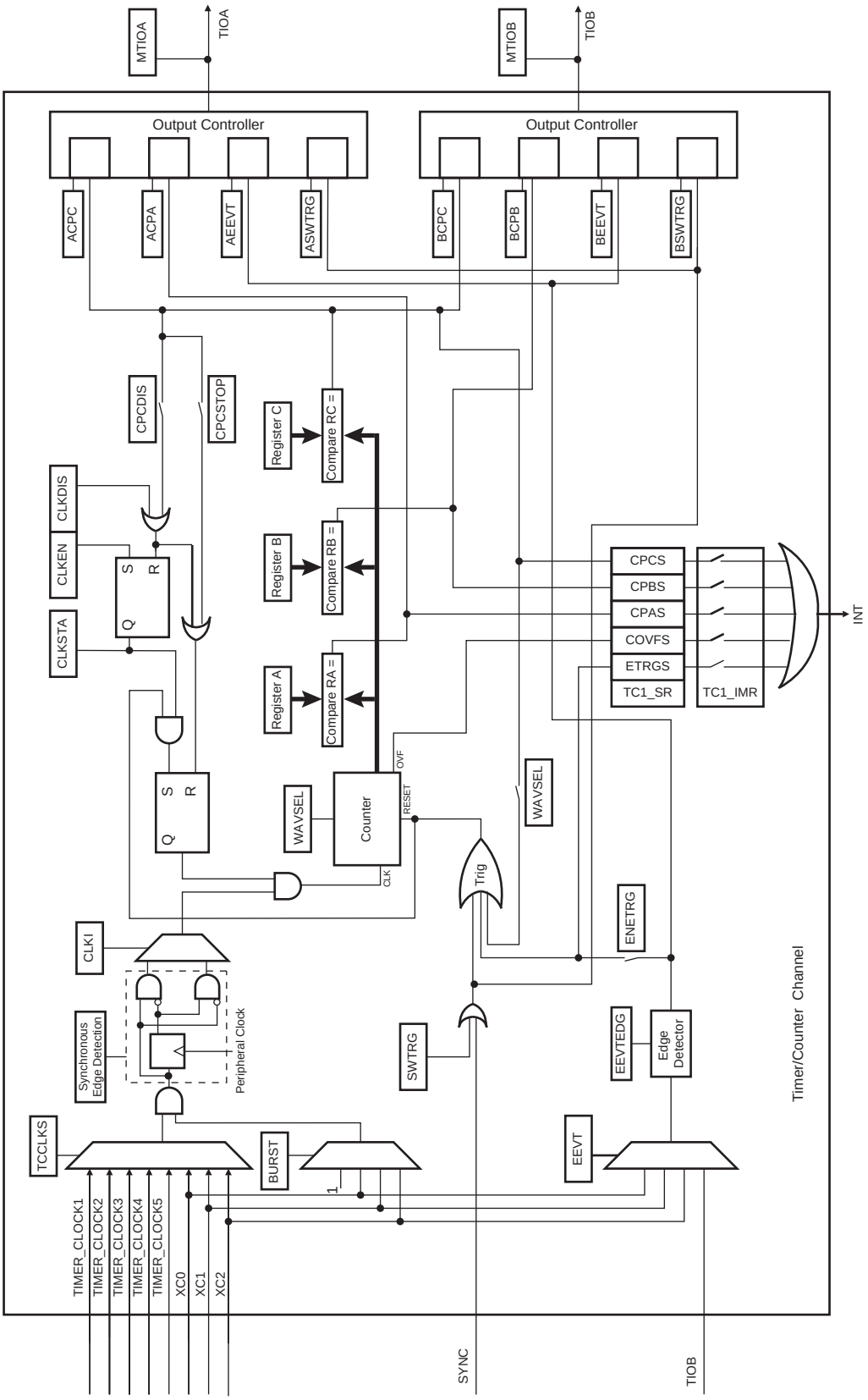
Depending on the WAVSEL parameter in TC_CMR, the behavior of TC_CV varies.

With any selection, TC_RA, TC_RB and TC_RC can all be used as compare registers.

RA Compare is used to control the TIOA output, RB Compare is used to control the TIOB output (if correctly configured) and RC Compare is used to control TIOA and/or TIOB outputs.

Downloaded from Arrow.com.

Figure 48-7. Waveform Mode



Downloaded from Arrow.com.

48.6.12.1 WAVSEL = 00

When WAVSEL = 00, the value of TC_CV is incremented from 0 to 2^{16}-1. Once 2^{16}-1 has been reached, the value of TC_CV is reset. Incrementation of TC_CV starts again and the cycle continues. See Figure 48-8.

An external event trigger or a software trigger can reset the value of TC_CV. It is important to note that the trigger may occur at any time. See Figure 48-9.

RC Compare cannot be programmed to generate a trigger in this configuration. At the same time, RC Compare can stop the counter clock (CPCSTOP = 1 in TC_CMR) and/or disable the counter clock (CPCDIS = 1 in TC_CMR).

Figure 48-8. WAVSEL = 00 without Trigger

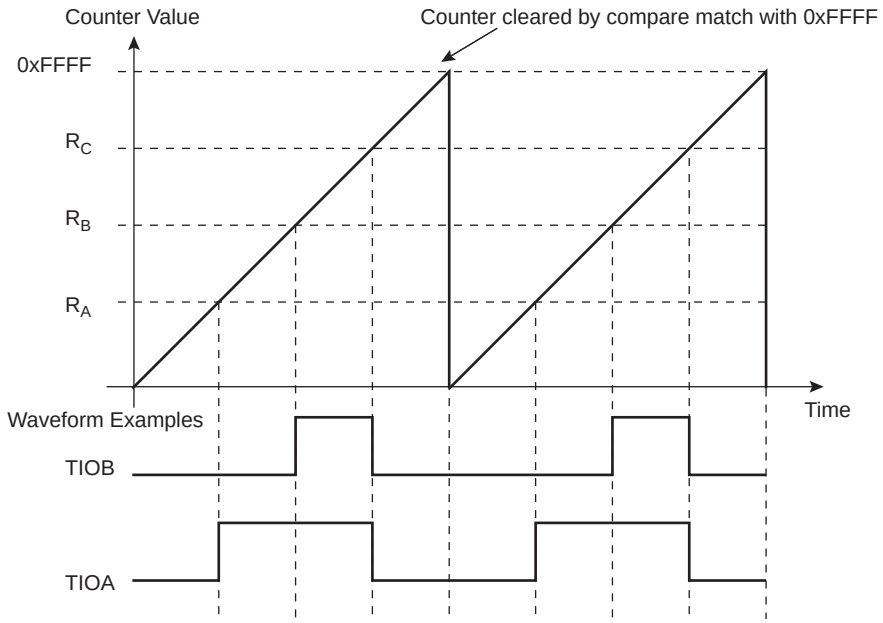
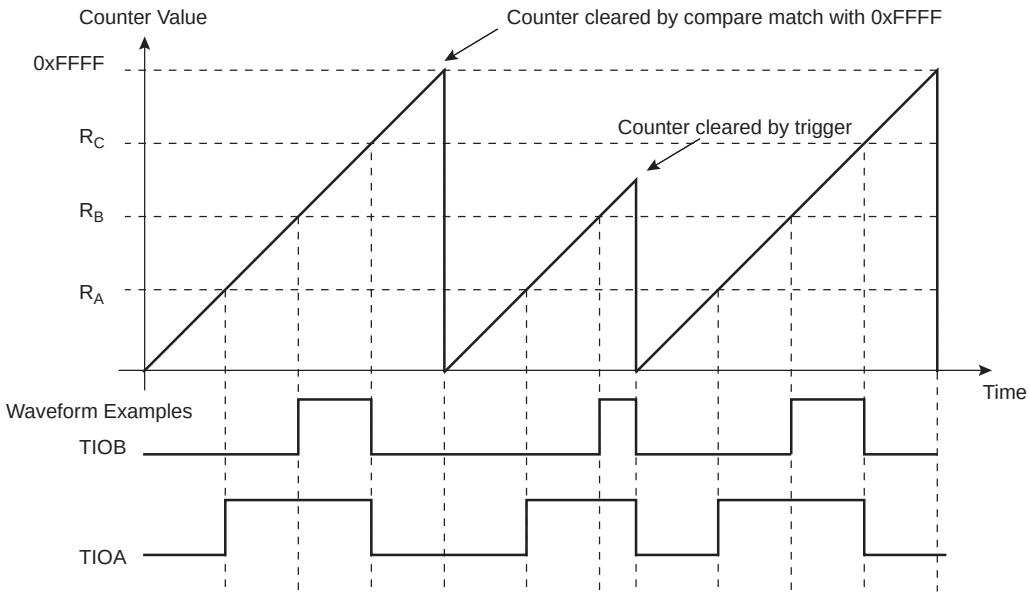


Figure 48-9. WAVSEL = 00 with Trigger



Downloaded from Arrow.com.

48.6.12.2 WAVSEL = 10

When WAVSEL = 10, the value of TC_CV is incremented from 0 to the value of RC, then automatically reset on a RC Compare. Once the value of TC_CV has been reset, it is then incremented and so on. See Figure 48-10.

It is important to note that TC_CV can be reset at any time by an external event or a software trigger if both are programmed correctly. See Figure 48-11.

In addition, RC Compare can stop the counter clock (CPCSTOP = 1 in TC_CMR) and/or disable the counter clock (CPCDIS = 1 in TC_CMR).

Figure 48-10. WAVSEL = 10 without Trigger

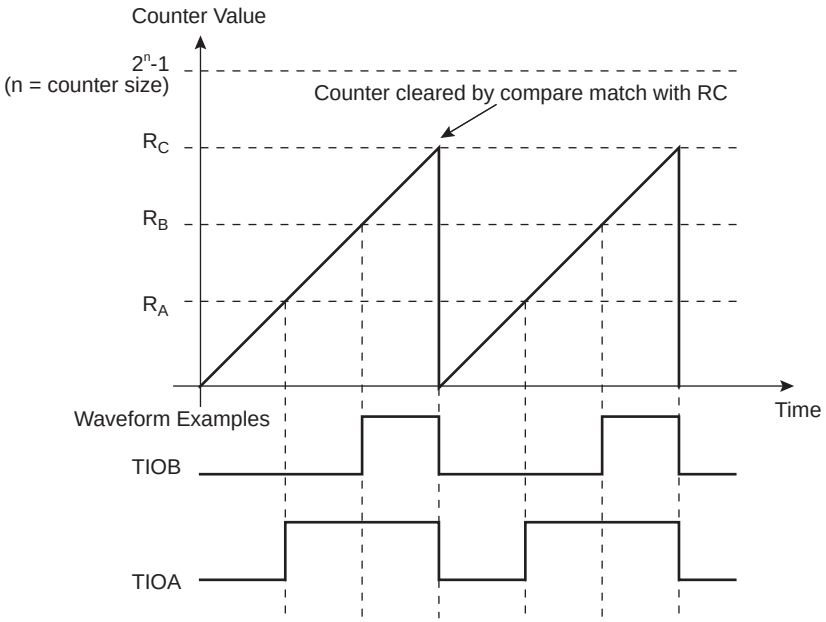
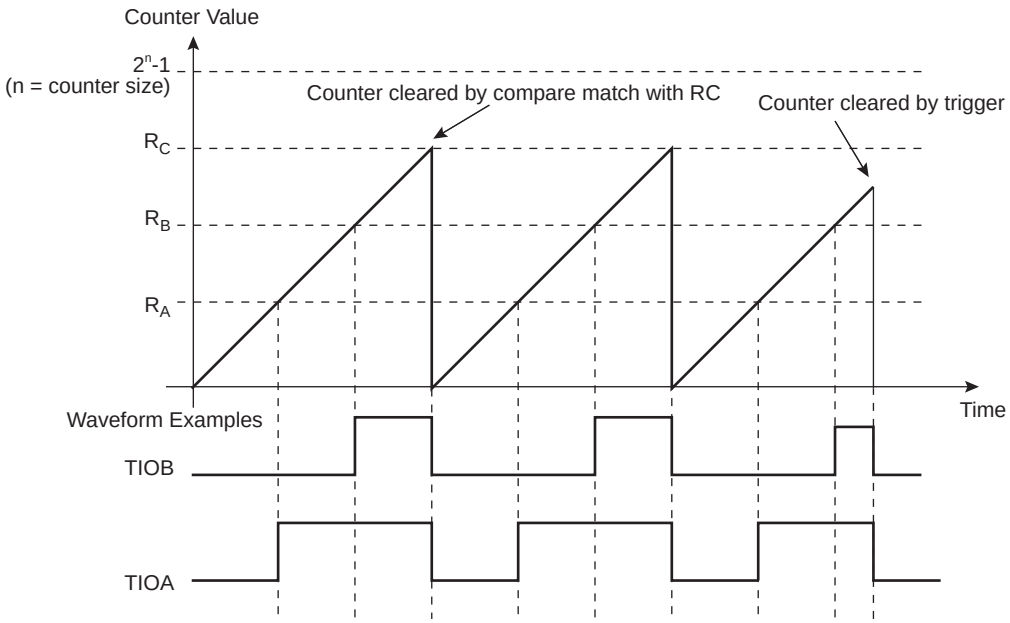


Figure 48-11. WAVSEL = 10 with Trigger



Downloaded from Arrow.com.

48.6.12.3 WAVSEL = 01

When WAVSEL = 01, the value of TC_CV is incremented from 0 to 2^{16}-1 . Once 2^{16}-1 is reached, the value of TC_CV is decremented to 0, then re-incremented to 2^{16}-1 and so on. See Figure 48-12.

A trigger such as an external event or a software trigger can modify TC_CV at any time. If a trigger occurs while TC_CV is incrementing, TC_CV then decrements. If a trigger is received while TC_CV is decrementing, TC_CV then increments. See Figure 48-13.

RC Compare cannot be programmed to generate a trigger in this configuration.

At the same time, RC Compare can stop the counter clock (CPCSTOP = 1) and/or disable the counter clock (CPCDIS = 1).

Figure 48-12. WAVSEL = 01 without Trigger

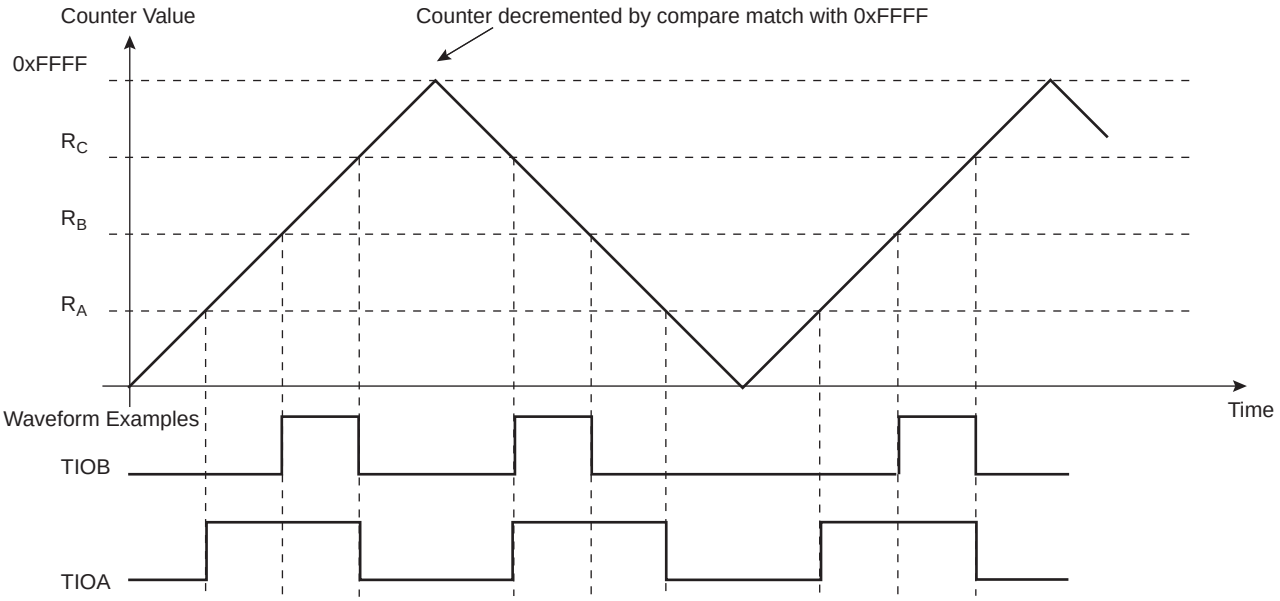
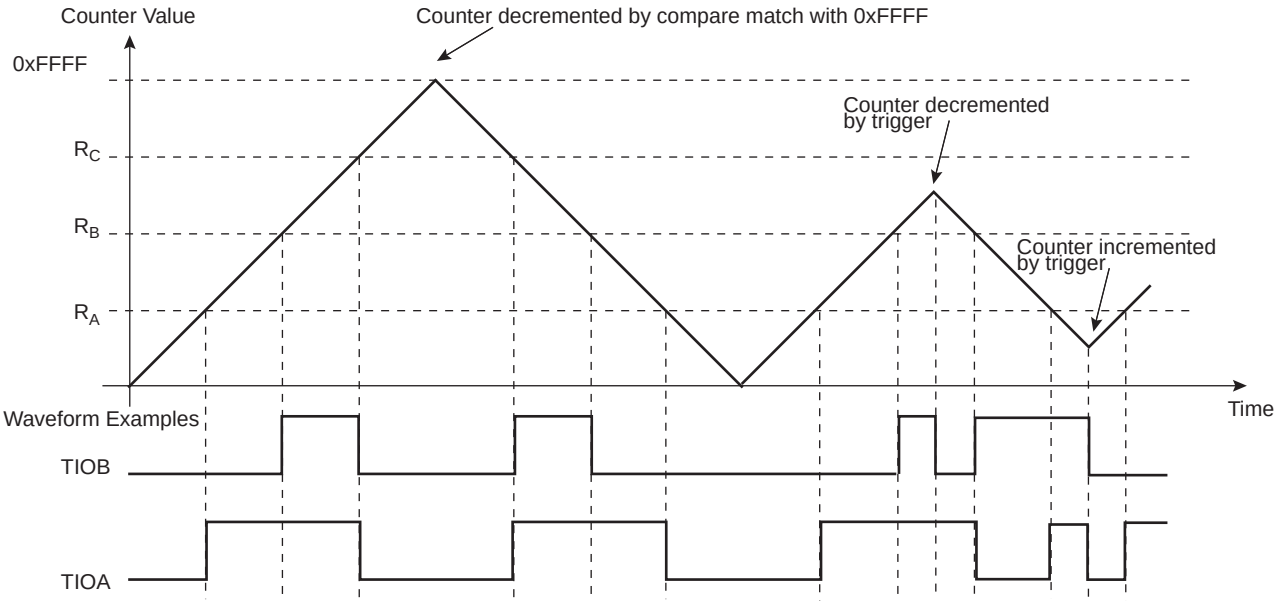


Figure 48-13. WAVSEL = 01 with Trigger



Downloaded from Arrow.com.

48.6.12.4 WAVSEL = 11

When WAVSEL = 11, the value of TC_CV is incremented from 0 to RC. Once RC is reached, the value of TC_CV is decremented to 0, then re-incremented to RC and so on. See Figure 48-14.

A trigger such as an external event or a software trigger can modify TC_CV at any time. If a trigger occurs while TC_CV is incrementing, TC_CV then decrements. If a trigger is received while TC_CV is decrementing, TC_CV then increments. See Figure 48-15.

RC Compare can stop the counter clock (CPCSTOP = 1) and/or disable the counter clock (CPCDIS = 1).

Figure 48-14. WAVSEL = 11 without Trigger

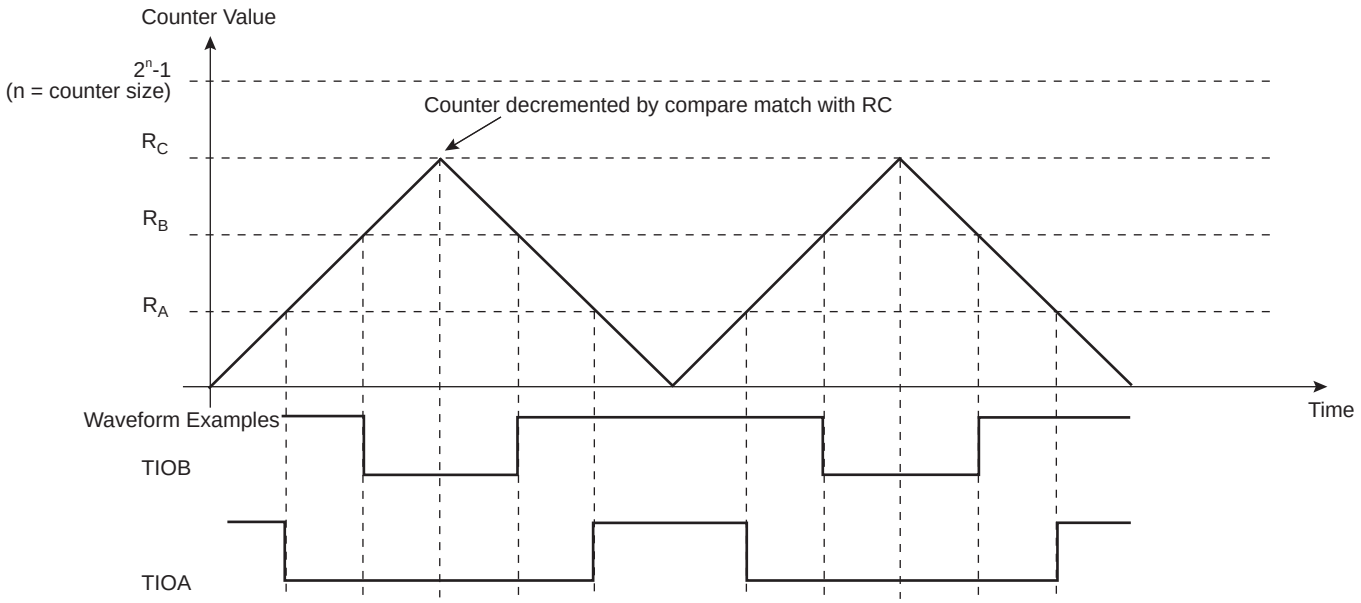
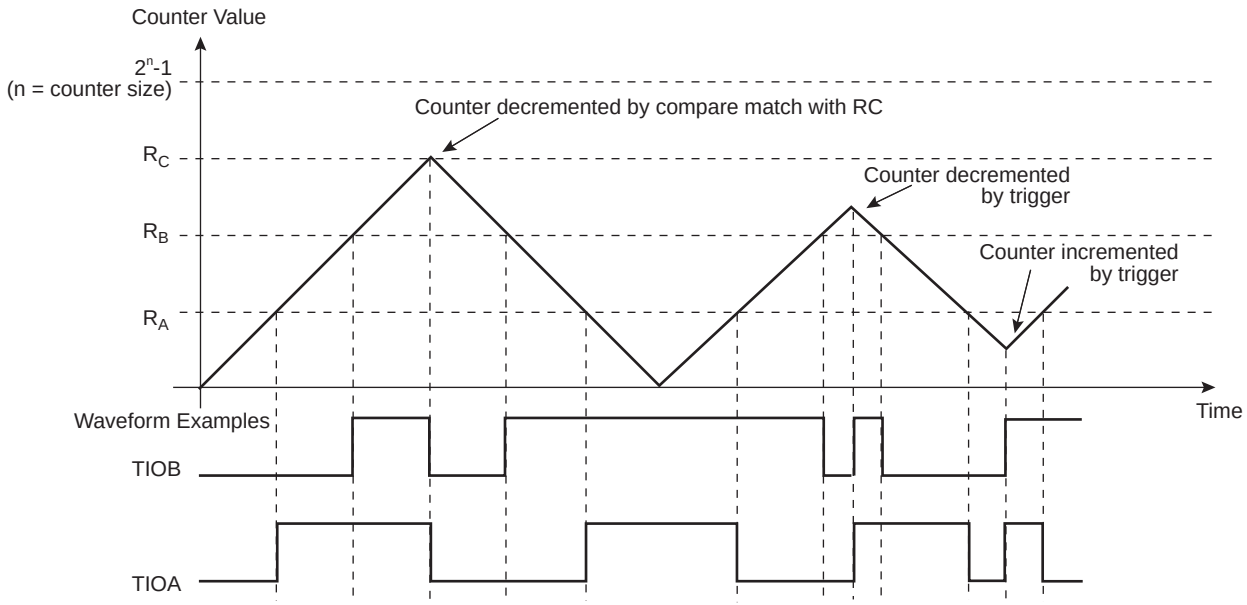


Figure 48-15. WAVSEL = 11 with Trigger



Downloaded from Arrow.com.

48.6.13 External Event/Trigger Conditions

An external event can be programmed to be detected on one of the clock sources (XC0, XC1, XC2) or TIOB. The external event selected can then be used as a trigger.

The EEVT parameter in TC_CMR selects the external trigger. The EEVTEDG parameter defines the trigger edge for each of the possible external triggers (rising, falling or both). If EEVTEDG is cleared (none), no external event is defined.

If TIOB is defined as an external event signal (EEVT = 0), TIOB is no longer used as an output and the compare register B is not used to generate waveforms and subsequently no IRQs. In this case the TC channel can only generate a waveform on TIOA.

When an external event is defined, it can be used as a trigger by setting bit ENETRG in the TC_CMR.

As in Capture mode, the SYNC signal and the software trigger are also available as triggers. RC Compare can also be used as a trigger depending on the parameter WAVSEL.

48.6.14 Synchronization with PWM

The inputs TIOA/B[2:0] can be bypassed, and thus channel trigger/capture events can be directly driven by the independent PWM module.

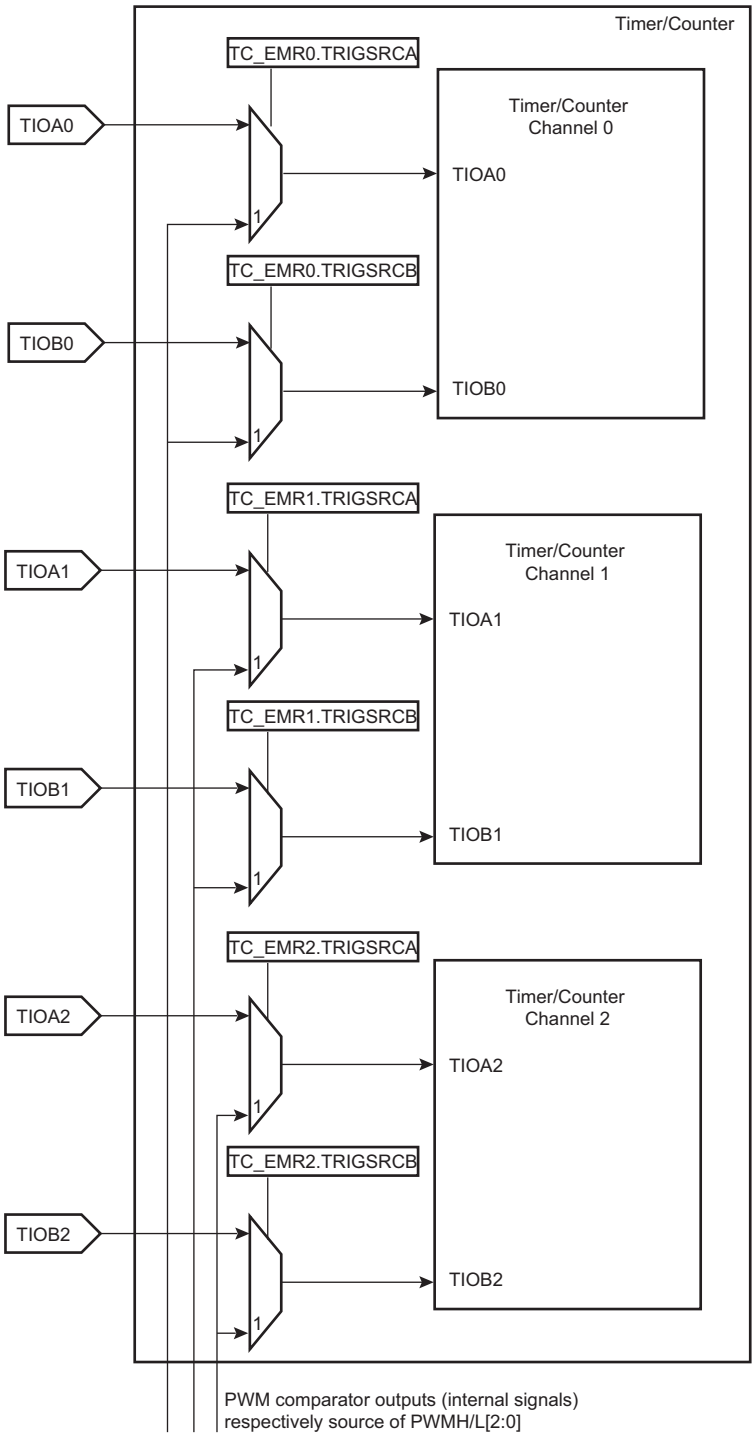
PWM comparator outputs (internal signals without dead-time insertion - OCx), respectively source of the PWMH/L[2:0] outputs, are routed to the internal TC inputs. These specific TC inputs are multiplexed with TIOA/B input signal to drive the internal trigger/capture events.

The selection can be programmed in the Extended Mode Register (TC_EMR) fields TRIGSRCA and TRIGSRCB (see Section 48.7.14 “TC Extended Mode Register”).

Each channel of the TC module can be synchronized by a different PWM channel as described in Figure 48-16.

Downloaded from Arrow.com.

Figure 48-16. Synchronization with PWM



Downloaded from Arrow.com.

48.6.15 Output Controller

The output controller defines the output level changes on TIOA and TIOB following an event. TIOB control is used only if TIOB is defined as output (not as an external event).

The following events control TIOA and TIOB: software trigger, external event and RC compare. RA compare controls TIOA and RB compare controls TIOB. Each of these events can be programmed to set, clear or toggle the output as defined in the corresponding parameter in TC_CMR.

48.6.16 Quadrature Decoder

48.6.16.1 Description

The quadrature decoder (QDEC) is driven by TIOA0, TIOB0, TIOB1 input pins and drives the timer/counter of channel 0 and 1. Channel 2 can be used as a time base in case of speed measurement requirements (refer to Figure 48-17).

When writing a 0 to bit QDEN of the TC_BMR, the QDEC is bypassed and the IO pins are directly routed to the timer counter function.

TIOA0 and TIOB0 are to be driven by the two dedicated quadrature signals from a rotary sensor mounted on the shaft of the off-chip motor.

A third signal from the rotary sensor can be processed through pin TIOB1 and is typically dedicated to be driven by an index signal if it is provided by the sensor. This signal is not required to decode the quadrature signals PHA, PHB.

Field TCCLKS of TC_CMRx must be configured to select XC0 input (i.e., 0x101). Field TC0XC0S has no effect as soon as the QDEC is enabled.

Either speed or position/revolution can be measured. Position channel 0 accumulates the edges of PHA, PHB input signals giving a high accuracy on motor position whereas channel 1 accumulates the index pulses of the sensor, therefore the number of rotations. Concatenation of both values provides a high level of precision on motion system position.

In Speed mode, position cannot be measured but revolution can be measured.

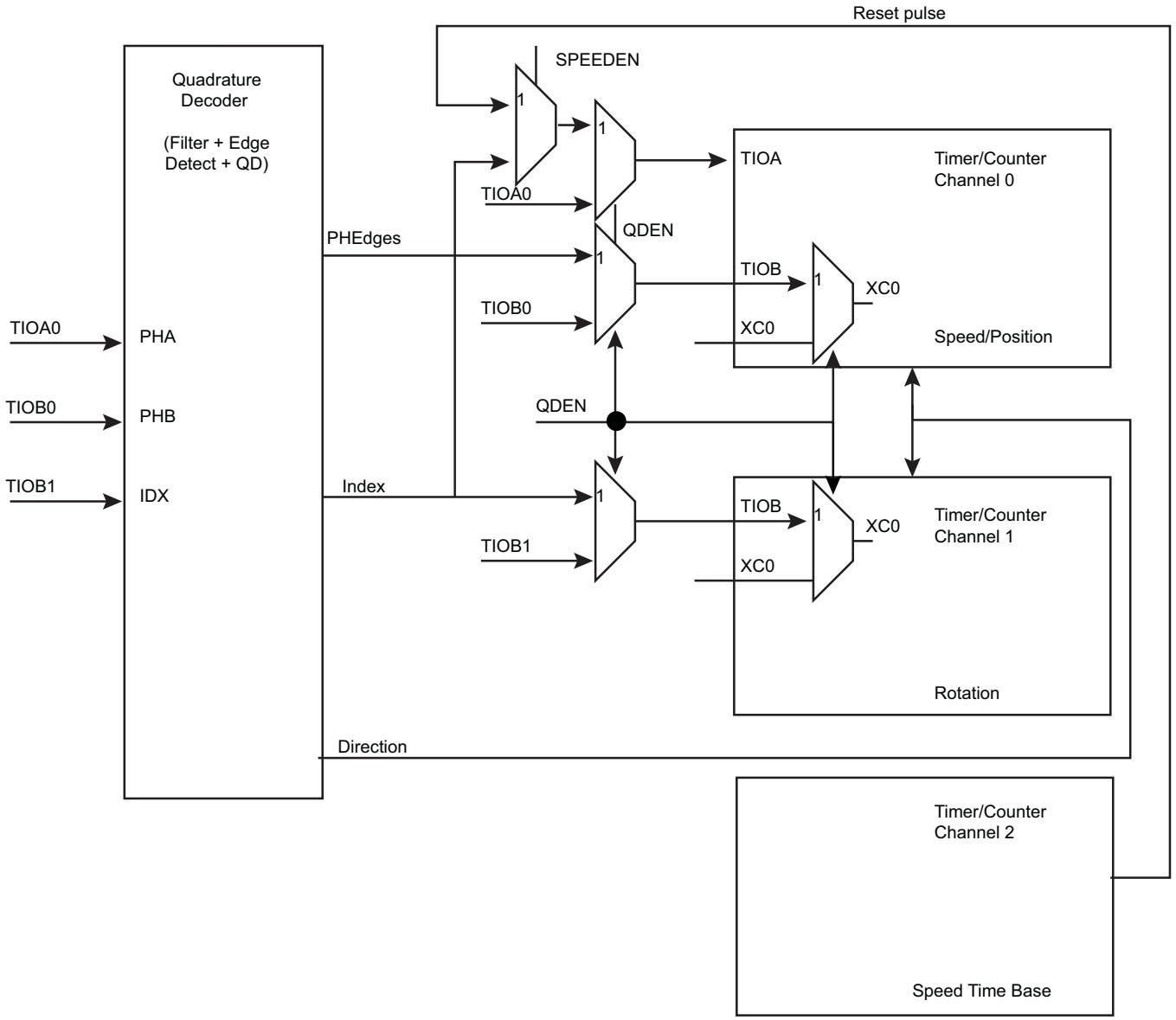
Inputs from the rotary sensor can be filtered prior to down-stream processing. Accommodation of input polarity, phase definition and other factors are configurable.

Interruptions can be generated on different events.

A compare function (using TC_RC) is available on channel 0 (speed/position) or channel 1 (rotation) and can generate an interrupt by means of the CPCS flag in the TC_SRx.

Downloaded from Arrow.com.

Figure 48-17. Predefined Connection of the Quadrature Decoder with Timer Counters



48.6.16.2 Input Pre-processing

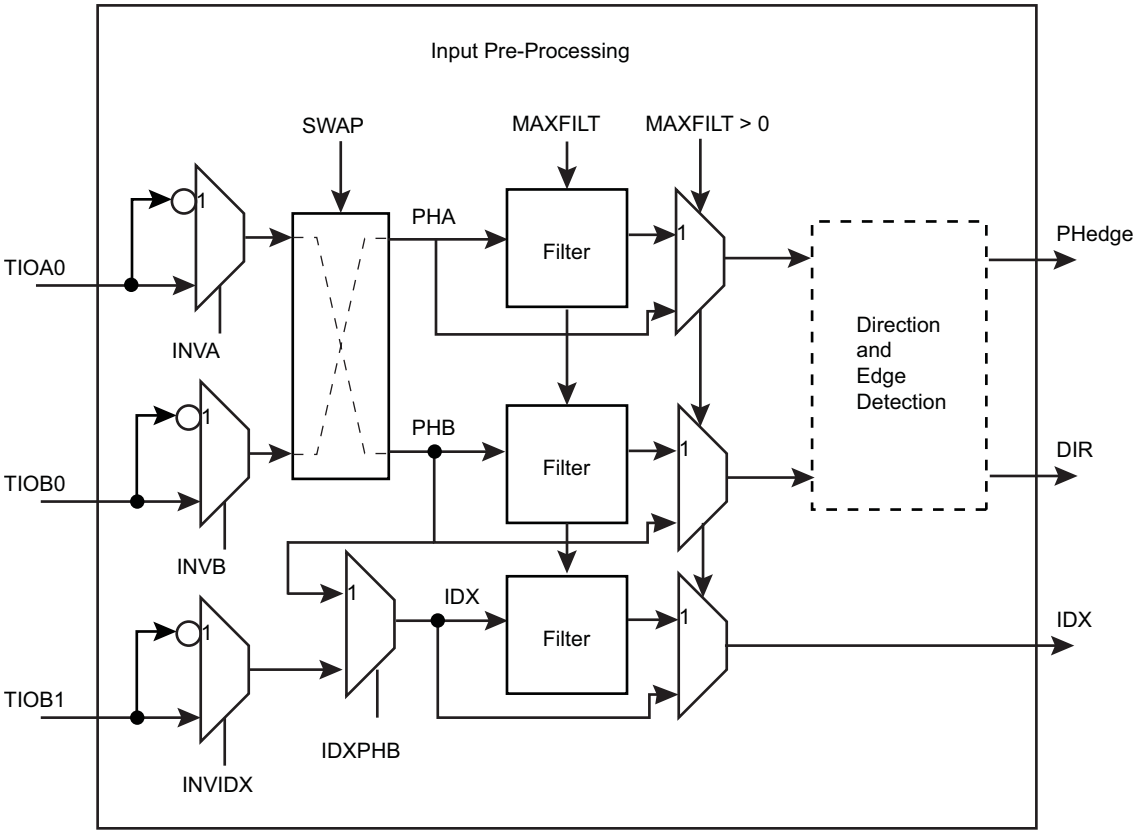
Input pre-processing consists of capabilities to take into account rotary sensor factors such as polarities and phase definition followed by configurable digital filtering.

Each input can be negated and swapping PHA, PHB is also configurable.

The MAXFILT field in the TC_BMR is used to configure a minimum duration for which the pulse is stated as valid. When the filter is active, pulses with a duration lower than MAXFILT +1 $\times$ $t_{peripheral\ clock}$ ns are not passed to down-stream logic.

Downloaded from Arrow.com.

Figure 48-18. Input Stage

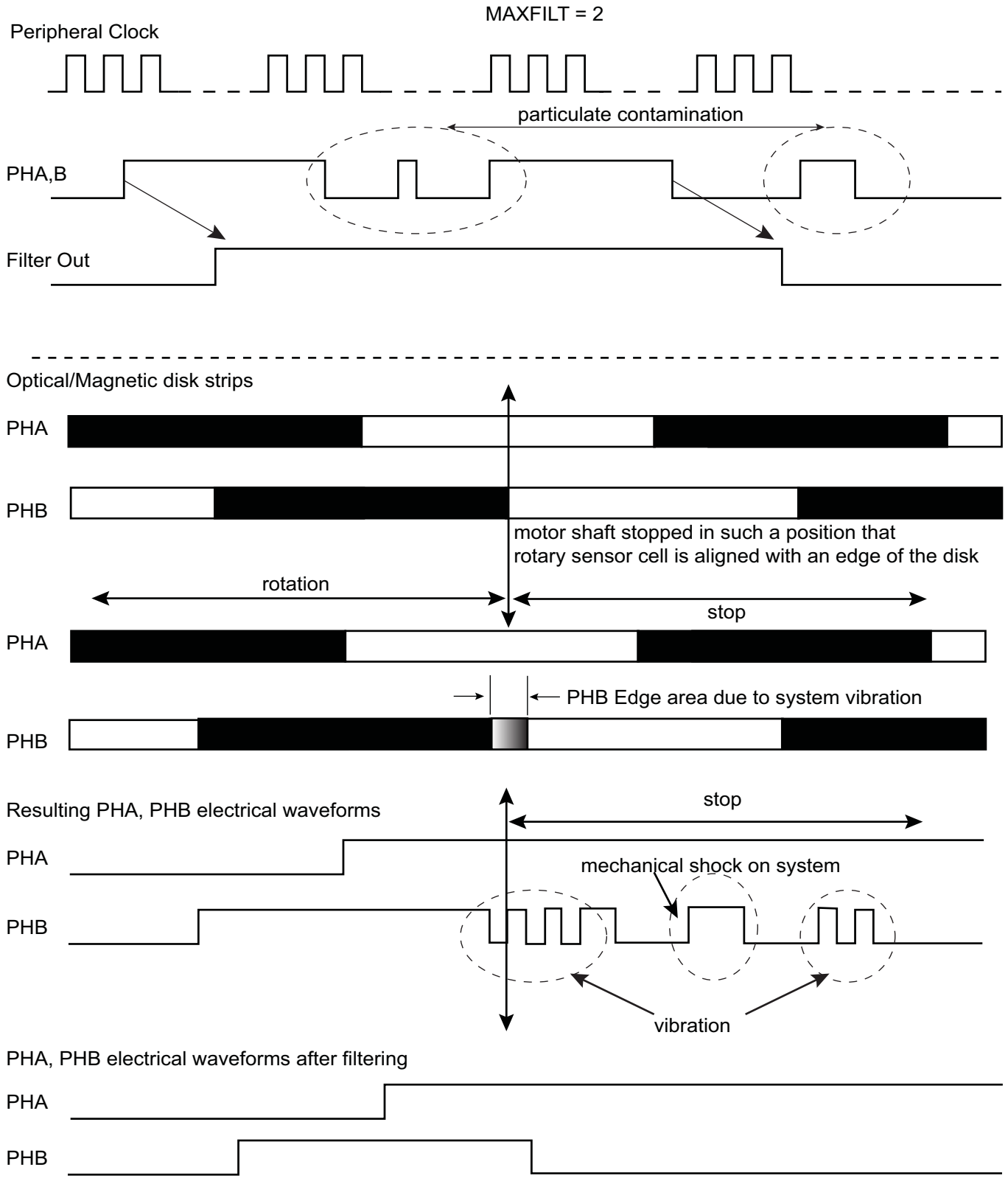


Input filtering can efficiently remove spurious pulses that might be generated by the presence of particulate contamination on the optical or magnetic disk of the rotary sensor.

Spurious pulses can also occur in environments with high levels of electro-magnetic interference. Or, simply if vibration occurs even when rotation is fully stopped and the shaft of the motor is in such a position that the beginning of one of the reflective or magnetic bars on the rotary sensor disk is aligned with the light or magnetic (Hall) receiver cell of the rotary sensor. Any vibration can make the PHA, PHB signals toggle for a short duration.

Downloaded from Arrow.com.

Figure 48-19. Filtering Examples



Downloaded from Arrow.com.

48.6.16.3 Direction Status and Change Detection

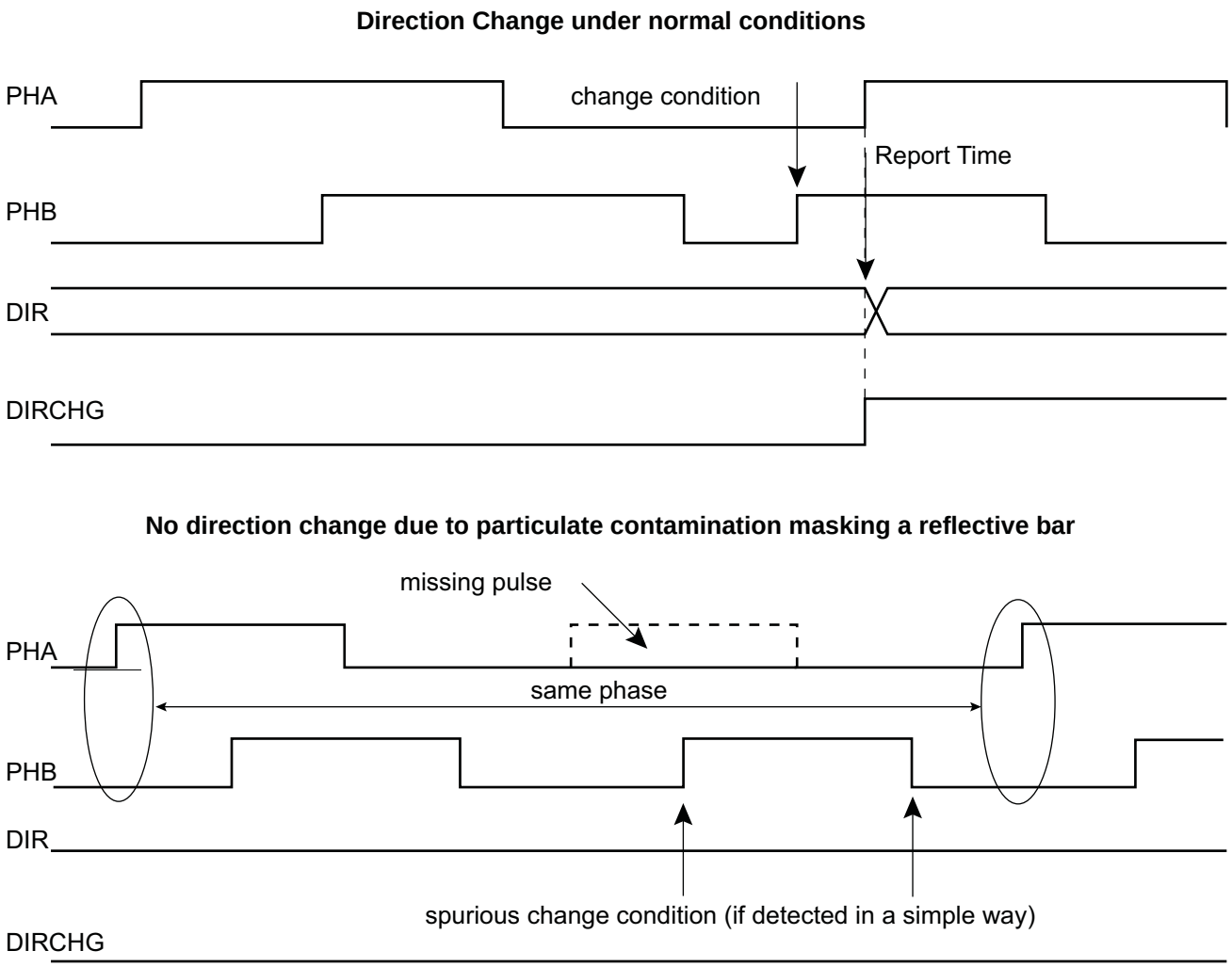
After filtering, the quadrature signals are analyzed to extract the rotation direction and edges of the two quadrature signals detected in order to be counted by timer/counter logic downstream.

The direction status can be directly read at anytime in the TC_QISR. The polarity of the direction flag status depends on the configuration written in TC_BMR. INVA, INVB, INVIDX, SWAP modify the polarity of DIR flag.

Any change in rotation direction is reported in the TC_QISR and can generate an interrupt.

The direction change condition is reported as soon as two consecutive edges on a phase signal have sampled the same value on the other phase signal and there is an edge on the other signal. The two consecutive edges of one phase signal sampling the same value on other phase signal is not sufficient to declare a direction change, for the reason that particulate contamination may mask one or more reflective bars on the optical or magnetic disk of the sensor. Refer to Figure 48-20 for waveforms.

Figure 48-20. Rotation Change Detection



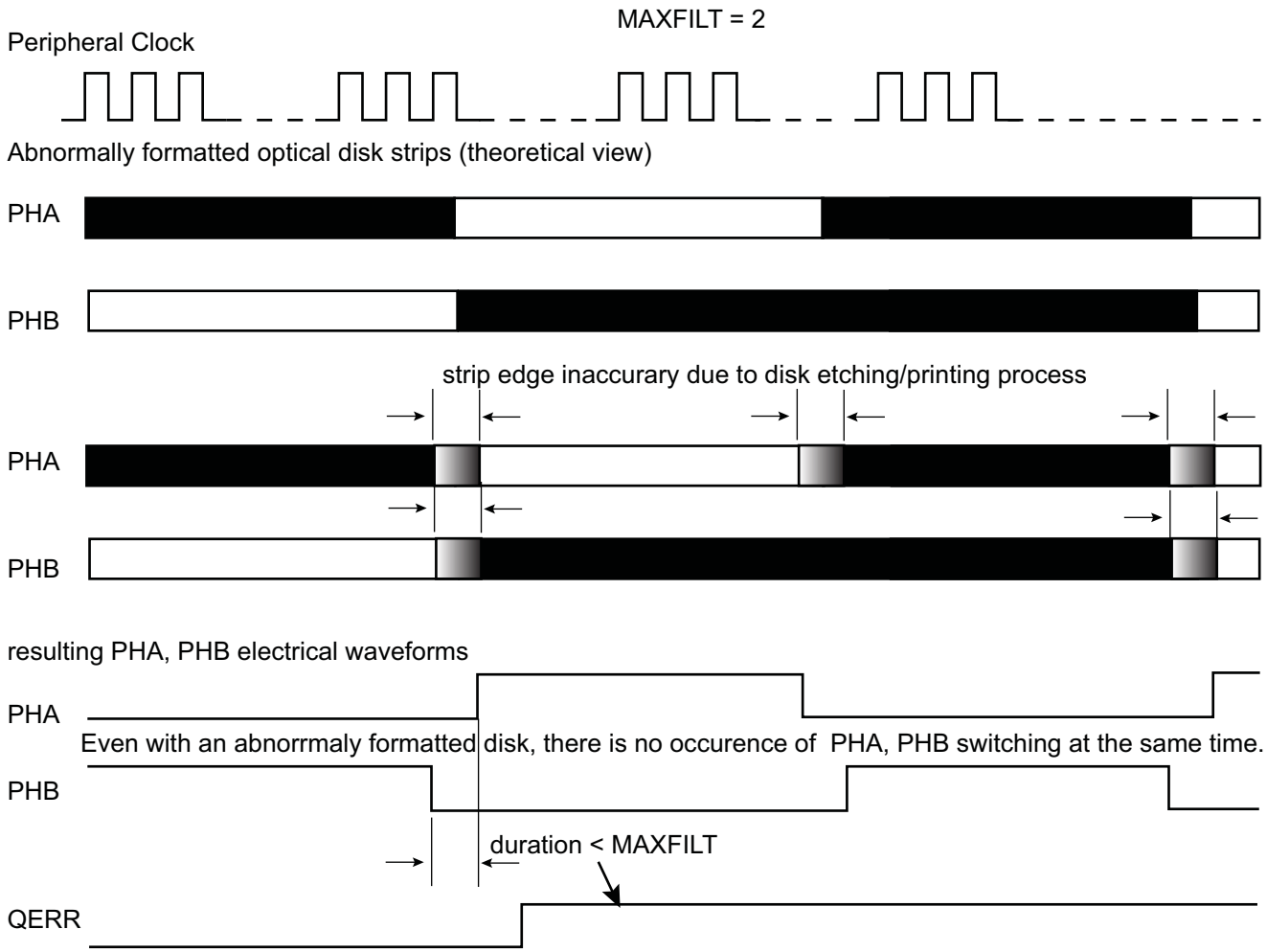
The direction change detection is disabled when QDTRANS is set in the TC_BMR. In this case, the DIR flag report must not be used.

A quadrature error is also reported by the QDEC via the QERR flag in the TC_QISR. This error is reported if the time difference between two edges on PHA, PHB is lower than a predefined value. This predefined value is

Downloaded from Arrow.com.

configurable and corresponds to (MAXFILT + 1) × $t_{peripheral\ clock}$ ns. After being filtered there is no reason to have two edges closer than (MAXFILT + 1) × $t_{peripheral\ clock}$ ns under normal mode of operation.

Figure 48-21. Quadrature Error Detection



MAXFILT must be tuned according to several factors such as the peripheral clock frequency, type of rotary sensor and rotation speed to be achieved.

48.6.16.4 Position and Rotation Measurement

When the POSEN bit is set in the TC_BMR, the motor axis position is processed on channel 0 (by means of the PHA, PHB edge detections) and the number of motor revolutions are recorded on channel 1 if the IDX signal is provided on the TIOB1 input. The position measurement can be read in the TC_CV0 register and the rotation measurement can be read in the TC_CV1 register.

Channel 0 and 1 must be configured in Capture mode (TC_CMR0.WAVE = 0). ‘Rising edge’ must be selected as the External Trigger Edge (TC_CMR.ETRGEDG = 0x01) and ‘TIOA’ must be selected as the External Trigger (TC_CMR.ABETRG = 0x1).

In parallel, the number of edges are accumulated on timer/counter channel 0 and can be read on the TC_CV0 register.

Therefore, the accurate position can be read on both TC_CV registers and concatenated to form a 32-bit word.

The timer/counter channel 0 is cleared for each increment of IDX count value.

Downloaded from Arrow.com.

Depending on the quadrature signals, the direction is decoded and allows to count up or down in timer/counter channels 0 and 1. The direction status is reported on TC_QISR.

48.6.16.5 Speed Measurement

When SPEEDEN is set in the TC_BMR, the speed measure is enabled on channel 0.

A time base must be defined on channel 2 by writing the TC_RC2 period register. Channel 2 must be configured in Waveform mode (WAVE bit set) in TC_CMR2. The WAVSEL field must be defined with 0x10 to clear the counter by comparison and matching with TC_RC value. Field ACPC must be defined at 0x11 to toggle TIOA output.

This time base is automatically fed back to TIOA of channel 0 when QDEN and SPEEDEN are set.

Channel 0 must be configured in Capture mode (WAVE = 0 in TC_CMR0). The ABETRG bit of TC_CMR0 must be configured at 1 to select TIOA as a trigger for this channel.

EDGTRG must be set to 0x01, to clear the counter on a rising edge of the TIOA signal and field LDRA must be set accordingly to 0x01, to load TC_RA0 at the same time as the counter is cleared (LDRB must be set to 0x01). As a consequence, at the end of each time base period the differentiation required for the speed calculation is performed.

The process must be started by configuring bits CLKEN and SWTRG in the TC_CCR.

The speed can be read on field RA in TC_RA0.

Channel 1 can still be used to count the number of revolutions of the motor.

48.6.17 2-bit Gray Up/Down Counter for Stepper Motor

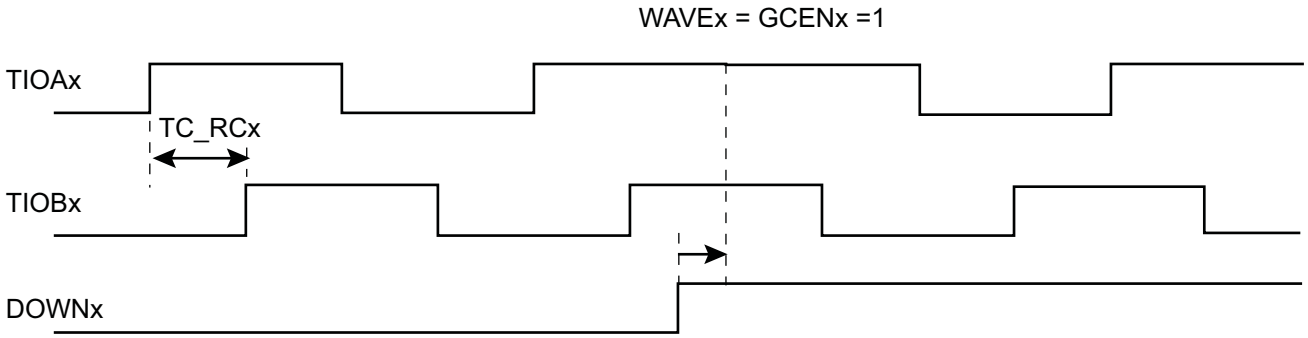
Each channel can be independently configured to generate a 2-bit Gray count waveform on corresponding TIOA, TIOB outputs by means of the GCEN bit in TC_SMMRx.

Up or Down count can be defined by writing bit DOWN in TC_SMMRx.

It is mandatory to configure the channel in Waveform mode in the TC_CMR.

The period of the counters can be programmed in TC_RCx.

Figure 48-22. 2-bit Gray Up/Down Counter



48.6.18 Fault Mode

At any time, the TC_RCx registers can be used to perform a comparison on the respective current channel counter value (TC_CVx) with the value of TC_RCx register.

The CPCSx flags can be set accordingly and an interrupt can be generated.

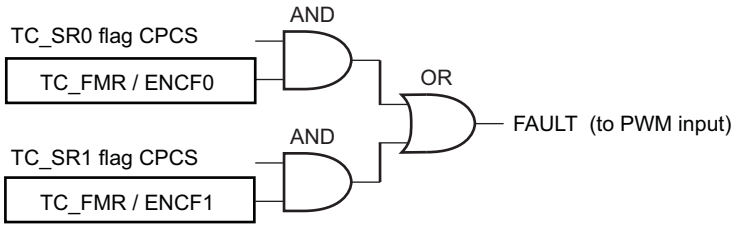
This interrupt is processed but requires an unpredictable amount of time to be achieve the required action.

It is possible to trigger the FAULT output of the TIMER1 with CPCS from TC_SR0 and/or CPCS from TC_SR1. Each source can be independently enabled/disabled in the TC_FMR.

Downloaded from Arrow.com.

This can be useful to detect an overflow on speed and/or position when QDEC is processed and to act immediately by using the FAULT output.

Figure 48-23. Fault Output Generation



48.6.19 Register Write Protection

To prevent any single software error from corrupting TC behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the TC Write Protection Mode Register (TC_WPMR).

The Timer Counter clock of the first channel must be enabled to access TC_WPMR.
The following registers can be write-protected:

- TC Block Mode Register
- TC Channel Mode Register: Capture Mode
- TC Channel Mode Register: Waveform Mode
- TC Fault Mode Register
- TC Stepper Motor Mode Register
- TC Register A
- TC Register B
- TC Register C
- TC Extended Mode Register

Downloaded from Arrow.com.

48.7 Timer Counter (TC) User Interface

Table 48-6. Register Mapping

Offset[1]	Register	Name	Access	Reset
0x00 + channel * 0x40 + 0x00	Channel Control Register	TC_CCR	Write-only	–
0x00 + channel * 0x40 + 0x04	Channel Mode Register	TC_CMR	Read/Write	0
0x00 + channel * 0x40 + 0x08	Stepper Motor Mode Register	TC_SMMR	Read/Write	0
0x00 + channel * 0x40 + 0x0C	Register AB	TC_RAB	Read-only	0
0x00 + channel * 0x40 + 0x10	Counter Value	TC_CV	Read-only	0
0x00 + channel * 0x40 + 0x14	Register A	TC_RA	Read/Write[2]	0
0x00 + channel * 0x40 + 0x18	Register B	TC_RB	Read/Write[2]	0
0x00 + channel * 0x40 + 0x1C	Register C	TC_RC	Read/Write	0
0x00 + channel * 0x40 + 0x20	Status Register	TC_SR	Read-only	0
0x00 + channel * 0x40 + 0x24	Interrupt Enable Register	TC_IER	Write-only	–
0x00 + channel * 0x40 + 0x28	Interrupt Disable Register	TC_IDR	Write-only	–
0x00 + channel * 0x40 + 0x2C	Interrupt Mask Register	TC_IMR	Read-only	0
0x00 + channel * 0x40 + 0x30	Extended Mode Register	TC_EMR	Read/Write	0
0xC0	Block Control Register	TC_BCR	Write-only	–
0xC4	Block Mode Register	TC_BMR	Read/Write	0
0xC8	QDEC Interrupt Enable Register	TC_QIER	Write-only	–
0xCC	QDEC Interrupt Disable Register	TC_QIDR	Write-only	–
0xD0	QDEC Interrupt Mask Register	TC_QIMR	Read-only	0
0xD4	QDEC Interrupt Status Register	TC_QISR	Read-only	0
0xD8	Fault Mode Register	TC_FMR	Read/Write	0
0xE4	Write Protection Mode Register	TC_WPMR	Read/Write	0
0xE8–0xFC	Reserved	–	–	–

Notes: 1. Channel index ranges from 0 to 2.
2. Read-only if TC_CMRx.WAVE = 0

Downloaded from Arrow.com.

48.7.1 TC Channel Control Register

Name: TC_CCRx [x=0..2]

Address: 0x4000C000 (0)[0], 0x4000C040 (0)[1], 0x4000C080 (0)[2], 0x40010000 (1)[0], 0x40010040 (1)[1], 0x40010080 (1)[2], 0x40014000 (2)[0], 0x40014040 (2)[1], 0x40014080 (2)[2], 0x40054000 (3)[0], 0x40054040 (3)[1], 0x40054080 (3)[2]

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	SWTRG	CLKDIS	CLKEN

- **CLKEN: Counter Clock Enable Command**

0: No effect.

1: Enables the clock if CLKDIS is not 1.

- **CLKDIS: Counter Clock Disable Command**

0: No effect.

1: Disables the clock.

- **SWTRG: Software Trigger Command**

0: No effect.

1: A software trigger is performed: the counter is reset and the clock is started.

Downloaded from Arrow.com.

48.7.2 TC Channel Mode Register: Capture Mode

Name: TC_CMRx [x=0..2] (CAPTURE_MODE)

Address: 0x4000C004 (0)[0], 0x4000C044 (0)[1], 0x4000C084 (0)[2], 0x40010004 (1)[0], 0x40010044 (1)[1], 0x40010084 (1)[2], 0x40014004 (2)[0], 0x40014044 (2)[1], 0x40014084 (2)[2], 0x40054004 (3)[0], 0x40054044 (3)[1], 0x40054084 (3)[2]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	SBSMPLR			LDRB		LDRA	

15	14	13	12	11	10	9	8
WAVE	CPCTRG	–	–	–	ABETRG	ETRGEDG	

7	6	5	4	3	2	1	0
LDBDIS	LDBSTOP	BURST		CLKI	TCCLKS		

This register can only be written if the WPEN bit is cleared in the TC Write Protection Mode Register.

- **TCCLKS: Clock Selection**

Value	Name	Description
0	TIMER_CLOCK1	Clock selected: internal PCK6 clock signal (from PMC)
1	TIMER_CLOCK2	Clock selected: internal MCK/8 clock signal (from PMC)
2	TIMER_CLOCK3	Clock selected: internal MCK/32 clock signal (from PMC)
3	TIMER_CLOCK4	Clock selected: internal MCK/128 clock signal (from PMC)
4	TIMER_CLOCK5	Clock selected: internal SLCK clock signal (from PMC)
5	XC0	Clock selected: XC0
6	XC1	Clock selected: XC1
7	XC2	Clock selected: XC2

To operate at maximum peripheral clock frequency, please refer to Section 48.7.14 “TC Extended Mode Register”.

- **CLKI: Clock Invert**

0: Counter is incremented on rising edge of the clock.

1: Counter is incremented on falling edge of the clock.

- **BURST: Burst Signal Selection**

Value	Name	Description
0	NONE	The clock is not gated by an external signal.
1	XC0	XC0 is ANDed with the selected clock.
2	XC1	XC1 is ANDed with the selected clock.
3	XC2	XC2 is ANDed with the selected clock.

Downloaded from Arrow.com.

- **LDBSTOP: Counter Clock Stopped with RB Loading**

0: Counter clock is not stopped when RB loading occurs.

1: Counter clock is stopped when RB loading occurs.

- **LDBDIS: Counter Clock Disable with RB Loading**

0: Counter clock is not disabled when RB loading occurs.

1: Counter clock is disabled when RB loading occurs.

- **ETRGEDG: External Trigger Edge Selection**

Value	Name	Description
0	NONE	The clock is not gated by an external signal.
1	RISING	Rising edge
2	FALLING	Falling edge
3	EDGE	Each edge

- **ABETRG: TIOA or TIOB External Trigger Selection**

0: TIOB is used as an external trigger.

1: TIOA is used as an external trigger.

- **CPCTRG: RC Compare Trigger Enable**

0: RC Compare has no effect on the counter and its clock.

1: RC Compare resets the counter and starts the counter clock.

- **WAVE: Waveform Mode**

0: Capture mode is enabled.

1: Capture mode is disabled (Waveform mode is enabled).

- **LDRA: RA Loading Edge Selection**

Value	Name	Description
0	NONE	None
1	RISING	Rising edge of TIOA
2	FALLING	Falling edge of TIOA
3	EDGE	Each edge of TIOA

- **LDRB: RB Loading Edge Selection**

Value	Name	Description
0	NONE	None
1	RISING	Rising edge of TIOA
2	FALLING	Falling edge of TIOA
3	EDGE	Each edge of TIOA

Downloaded from Arrow.com.

- **SBSMPLR: Loading Edge Subsampling Ratio**

Value	Name	Description
0	ONE	Load a Capture Register each selected edge
1	HALF	Load a Capture Register every 2 selected edges
2	FOURTH	Load a Capture Register every 4 selected edges
3	EIGHTH	Load a Capture Register every 8 selected edges
4	SIXTEENTH	Load a Capture Register every 16 selected edges

Downloaded from Arrow.com.

48.7.3 TC Channel Mode Register: Waveform Mode

Name: TC_CMRx [x=0..2] (WAVEFORM_MODE)

Address: 0x4000C004 (0)[0], 0x4000C044 (0)[1], 0x4000C084 (0)[2], 0x40010004 (1)[0], 0x40010044 (1)[1], 0x40010084 (1)[2], 0x40014004 (2)[0], 0x40014044 (2)[1], 0x40014084 (2)[2], 0x40054004 (3)[0], 0x40054044 (3)[1], 0x40054084 (3)[2]

Access: Read/Write

31	30	29	28	27	26	25	24
BSWTRG		BEEVT		BCPC		BCPB	
23	22	21	20	19	18	17	16
ASWTRG		AEEVT		ACPC		ACPA	
15	14	13	12	11	10	9	8
WAVE	WAVSEL		ENETRG	EEVT		EEVTEDG	
7	6	5	4	3	2	1	0
CPCDIS	CPCSTOP	BURST		CLKI	TCCLKS		

This register can only be written if the WPEN bit is cleared in the TC Write Protection Mode Register.

- **TCCLKS: Clock Selection**

Value	Name	Description
0	TIMER_CLOCK1	Clock selected: internal PCK6 clock signal (from PMC)
1	TIMER_CLOCK2	Clock selected: internal MCK/8 clock signal (from PMC)
2	TIMER_CLOCK3	Clock selected: internal MCK/32 clock signal (from PMC)
3	TIMER_CLOCK4	Clock selected: internal MCK/128 clock signal (from PMC)
4	TIMER_CLOCK5	Clock selected: internal SLCK clock signal (from PMC)
5	XC0	Clock selected: XC0
6	XC1	Clock selected: XC1
7	XC2	Clock selected: XC2

To operate at maximum peripheral clock frequency, please refer to Section 48.7.14 “TC Extended Mode Register”.

- **CLKI: Clock Invert**

0: Counter is incremented on rising edge of the clock.

1: Counter is incremented on falling edge of the clock.

- **BURST: Burst Signal Selection**

Value	Name	Description
0	NONE	The clock is not gated by an external signal.
1	XC0	XC0 is ANDed with the selected clock.
2	XC1	XC1 is ANDed with the selected clock.
3	XC2	XC2 is ANDed with the selected clock.

Downloaded from Arrow.com.

- **CPCSTOP: Counter Clock Stopped with RC Compare**

0: Counter clock is not stopped when counter reaches RC.

1: Counter clock is stopped when counter reaches RC.

- **CPCDIS: Counter Clock Disable with RC Compare**

0: Counter clock is not disabled when counter reaches RC.

1: Counter clock is disabled when counter reaches RC.

- **EEVTEDG: External Event Edge Selection**

Value	Name	Description
0	NONE	None
1	RISING	Rising edge
2	FALLING	Falling edge
3	EDGE	Each edge

- **EEVT: External Event Selection**

Signal selected as external event.

Value	Name	Description	TIOB Direction
0	TIOB	TIOB(1)	Input
1	XC0	XC0	Output
2	XC1	XC1	Output
3	XC2	XC2	Output

Note: 1. If TIOB is chosen as the external event signal, it is configured as an input and no longer generates waveforms and subsequently no IRQs.

- **ENETRG: External Event Trigger Enable**

0: The external event has no effect on the counter and its clock.

1: The external event resets the counter and starts the counter clock.

Note: Whatever the value programmed in ENETRG, the selected external event only controls the TIOA output and TIOB if not used as input (trigger event input or other input used).

- **WAVSEL: Waveform Selection**

Value	Name	Description
0	UP	UP mode without automatic trigger on RC Compare
1	UPDOWN	UPDOWN mode without automatic trigger on RC Compare
2	UP_RC	UP mode with automatic trigger on RC Compare
3	UPDOWN_RC	UPDOWN mode with automatic trigger on RC Compare

- **WAVE: Waveform Mode**

0: Waveform mode is disabled (Capture mode is enabled).

1: Waveform mode is enabled.

Downloaded from Arrow.com.

- **ACPA: RA Compare Effect on TIOA**

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

- **ACPC: RC Compare Effect on TIOA**

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

- **AEEVT: External Event Effect on TIOA**

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

- **ASWTRG: Software Trigger Effect on TIOA**

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

- **BCPB: RB Compare Effect on TIOB**

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

- **BCPC: RC Compare Effect on TIOB**

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

Downloaded from Arrow.com.

- **BEEVT: External Event Effect on TIOB**

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

- **BSWTRG: Software Trigger Effect on TIOB**

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

Downloaded from Arrow.com.

48.7.4 TC Stepper Motor Mode Register

Name: TC_SMMRx [x=0..2]

Address: 0x4000C008 (0)[0], 0x4000C048 (0)[1], 0x4000C088 (0)[2], 0x40010008 (1)[0], 0x40010048 (1)[1], 0x40010088 (1)[2], 0x40014008 (2)[0], 0x40014048 (2)[1], 0x40014088 (2)[2], 0x40054008 (3)[0], 0x40054048 (3)[1], 0x40054088 (3)[2]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	DOWN	GCEN

This register can only be written if the WPEN bit is cleared in the TC Write Protection Mode Register.

- **GCEN: Gray Count Enable**

0: TIOAx [x=0..2] and TIOBx [x=0..2] are driven by internal counter of channel x.

1: TIOAx [x=0..2] and TIOBx [x=0..2] are driven by a 2-bit Gray counter.

- **DOWN: Down Count**

0: Up counter.

1: Down counter.

Downloaded from Arrow.com.

48.7.5 TC Register AB

Name: TC_RABx [x=0..2]

Address: 0x4000C00C (0)[0], 0x4000C04C (0)[1], 0x4000C08C (0)[2], 0x4001000C (1)[0], 0x4001004C (1)[1], 0x4001008C (1)[2], 0x4001400C (2)[0], 0x4001404C (2)[1], 0x4001408C (2)[2], 0x4005400C (3)[0], 0x4005404C (3)[1], 0x4005408C (3)[2]

Access: Read-only

31	30	29	28	27	26	25	24
RAB							

23	22	21	20	19	18	17	16
RAB							

15	14	13	12	11	10	9	8
RAB							

7	6	5	4	3	2	1	0
RAB							

- **RAB: Register A or Register B**

RAB contains the next unread capture Register A or Register B value in real time. It is usually read by the DMA after a request due to a valid load edge on TIOA.

When DMA is used, the RAB register address must be configured as source address of the transfer.

Downloaded from Arrow.com.

48.7.6 TC Counter Value Register

Name: TC_CVx [x=0..2]

Address: 0x4000C010 (0)[0], 0x4000C050 (0)[1], 0x4000C090 (0)[2], 0x40010010 (1)[0], 0x40010050 (1)[1], 0x40010090 (1)[2], 0x40014010 (2)[0], 0x40014050 (2)[1], 0x40014090 (2)[2], 0x40054010 (3)[0], 0x40054050 (3)[1], 0x40054090 (3)[2]

Access: Read-only

31	30	29	28	27	26	25	24
CV							

23	22	21	20	19	18	17	16
CV							

15	14	13	12	11	10	9	8
CV							

7	6	5	4	3	2	1	0
CV							

- **CV: Counter Value**

CV contains the counter value in real time.

IMPORTANT: For 16-bit channels, CV field size is limited to register bits 15:0.

Downloaded from Arrow.com.

48.7.7 TC Register A

Name: TC_RAx [x=0..2]

Address: 0x4000C014 (0)[0], 0x4000C054 (0)[1], 0x4000C094 (0)[2], 0x40010014 (1)[0], 0x40010054 (1)[1], 0x40010094 (1)[2], 0x40014014 (2)[0], 0x40014054 (2)[1], 0x40014094 (2)[2], 0x40054014 (3)[0], 0x40054054 (3)[1], 0x40054094 (3)[2]

Access: Read-only if TC_CMRx.WAVE = 0, Read/Write if TC_CMRx.WAVE = 1

31	30	29	28	27	26	25	24
RA							

23	22	21	20	19	18	17	16
RA							

15	14	13	12	11	10	9	8
RA							

7	6	5	4	3	2	1	0
RA							

This register can only be written if the WPEN bit is cleared in the TC Write Protection Mode Register.

- **RA: Register A**

RA contains the Register A value in real time.

IMPORTANT: For 16-bit channels, RA field size is limited to register bits 15:0.

Downloaded from Arrow.com.

48.7.8 TC Register B

Name: TC_RBx [x=0..2]

Address: 0x4000C018 (0)[0], 0x4000C058 (0)[1], 0x4000C098 (0)[2], 0x40010018 (1)[0], 0x40010058 (1)[1], 0x40010098 (1)[2], 0x40014018 (2)[0], 0x40014058 (2)[1], 0x40014098 (2)[2], 0x40054018 (3)[0], 0x40054058 (3)[1], 0x40054098 (3)[2]

Access: Read-only if TC_CMRx.WAVE = 0, Read/Write if TC_CMRx.WAVE = 1

31	30	29	28	27	26	25	24
RB							
23	22	21	20	19	18	17	16
RB							
15	14	13	12	11	10	9	8
RB							
7	6	5	4	3	2	1	0
RB							

This register can only be written if the WPEN bit is cleared in the TC Write Protection Mode Register.

- **RB: Register B**

RB contains the Register B value in real time.

IMPORTANT: For 16-bit channels, RB field size is limited to register bits 15:0.

Downloaded from Arrow.com.

48.7.9 TC Register C

Name: TC_RCx [x=0..2]

Address: 0x4000C01C (0)[0], 0x4000C05C (0)[1], 0x4000C09C (0)[2], 0x4001001C (1)[0], 0x4001005C (1)[1], 0x4001009C (1)[2], 0x4001401C (2)[0], 0x4001405C (2)[1], 0x4001409C (2)[2], 0x4005401C (3)[0], 0x4005405C (3)[1], 0x4005409C (3)[2]

Access: Read/Write

31	30	29	28	27	26	25	24
RC							

23	22	21	20	19	18	17	16
RC							

15	14	13	12	11	10	9	8
RC							

7	6	5	4	3	2	1	0
RC							

This register can only be written if the WPEN bit is cleared in the TC Write Protection Mode Register.

- **RC: Register C**

RC contains the Register C value in real time.

IMPORTANT: For 16-bit channels, RC field size is limited to register bits 15:0.

Downloaded from Arrow.com.

48.7.10 TC Status Register

Name: TC_SRx [x=0..2]

Address: 0x4000C020 (0)[0], 0x4000C060 (0)[1], 0x4000C0A0 (0)[2], 0x40010020 (1)[0], 0x40010060 (1)[1], 0x400100A0 (1)[2], 0x40014020 (2)[0], 0x40014060 (2)[1], 0x400140A0 (2)[2], 0x40054020 (3)[0], 0x40054060 (3)[1], 0x400540A0 (3)[2]

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	MTIOB	MTIOA	CLKSTA
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: Counter Overflow Status (cleared on read)**

0: No counter overflow has occurred since the last read of the Status Register.

1: A counter overflow has occurred since the last read of the Status Register.

- **LOVRS: Load Overrun Status (cleared on read)**

0: Load overrun has not occurred since the last read of the Status Register or TC_CMRx.WAVE = 1.

1: RA or RB have been loaded at least twice without any read of the corresponding register since the last read of the Status Register, if TC_CMRx.WAVE = 0.

- **CPAS: RA Compare Status (cleared on read)**

0: RA Compare has not occurred since the last read of the Status Register or TC_CMRx.WAVE = 0.

1: RA Compare has occurred since the last read of the Status Register, if TC_CMRx.WAVE = 1.

- **CPBS: RB Compare Status (cleared on read)**

0: RB Compare has not occurred since the last read of the Status Register or TC_CMRx.WAVE = 0.

1: RB Compare has occurred since the last read of the Status Register, if TC_CMRx.WAVE = 1.

- **CPCS: RC Compare Status (cleared on read)**

0: RC Compare has not occurred since the last read of the Status Register.

1: RC Compare has occurred since the last read of the Status Register.

- **LDRAS: RA Loading Status (cleared on read)**

0: RA Load has not occurred since the last read of the Status Register or TC_CMRx.WAVE = 1.

1: RA Load has occurred since the last read of the Status Register, if TC_CMRx.WAVE = 0.

- **LDRBS: RB Loading Status (cleared on read)**

0: RB Load has not occurred since the last read of the Status Register or TC_CMRx.WAVE = 1.

1: RB Load has occurred since the last read of the Status Register, if TC_CMRx.WAVE = 0.

Downloaded from Arrow.com.

- **ETRGS: External Trigger Status (cleared on read)**

0: External trigger has not occurred since the last read of the Status Register.

1: External trigger has occurred since the last read of the Status Register.

- **CLKSTA: Clock Enabling Status**

0: Clock is disabled.

1: Clock is enabled.

- **MTIOA: TIOA Mirror**

0: TIOA is low. If TC_CMRx.WAVE = 0, this means that TIOA pin is low. If TC_CMRx.WAVE = 1, this means that TIOA is driven low.

1: TIOA is high. If TC_CMRx.WAVE = 0, this means that TIOA pin is high. If TC_CMRx.WAVE = 1, this means that TIOA is driven high.

- **MTIOB: TIOB Mirror**

0: TIOB is low. If TC_CMRx.WAVE = 0, this means that TIOB pin is low. If TC_CMRx.WAVE = 1, this means that TIOB is driven low.

1: TIOB is high. If TC_CMRx.WAVE = 0, this means that TIOB pin is high. If TC_CMRx.WAVE = 1, this means that TIOB is driven high.

Downloaded from Arrow.com.

48.7.11 TC Interrupt Enable Register

Name: TC_IERx [x=0..2]

Address: 0x4000C024 (0)[0], 0x4000C064 (0)[1], 0x4000C0A4 (0)[2], 0x40010024 (1)[0], 0x40010064 (1)[1], 0x400100A4 (1)[2], 0x40014024 (2)[0], 0x40014064 (2)[1], 0x400140A4 (2)[2], 0x40054024 (3)[0], 0x40054064 (3)[1], 0x400540A4 (3)[2]

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: Counter Overflow**

0: No effect.

1: Enables the Counter Overflow Interrupt.

- **LOVRS: Load Overrun**

0: No effect.

1: Enables the Load Overrun Interrupt.

- **CPAS: RA Compare**

0: No effect.

1: Enables the RA Compare Interrupt.

- **CPBS: RB Compare**

0: No effect.

1: Enables the RB Compare Interrupt.

- **CPCS: RC Compare**

0: No effect.

1: Enables the RC Compare Interrupt.

- **LDRAS: RA Loading**

0: No effect.

1: Enables the RA Load Interrupt.

- **LDRBS: RB Loading**

0: No effect.

1: Enables the RB Load Interrupt.

Downloaded from Arrow.com.

- **ETRGS: External Trigger**

0: No effect.

1: Enables the External Trigger Interrupt.

Downloaded from Arrow.com.

48.7.12 TC Interrupt Disable Register

Name: TC_IDRx [x=0..2]

Address: 0x4000C028 (0)[0], 0x4000C068 (0)[1], 0x4000C0A8 (0)[2], 0x40010028 (1)[0], 0x40010068 (1)[1], 0x400100A8 (1)[2], 0x40014028 (2)[0], 0x40014068 (2)[1], 0x400140A8 (2)[2], 0x40054028 (3)[0], 0x40054068 (3)[1], 0x400540A8 (3)[2]

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: Counter Overflow**

0: No effect.

1: Disables the Counter Overflow Interrupt.

- **LOVRS: Load Overrun**

0: No effect.

1: Disables the Load Overrun Interrupt (if TC_CMRx.WAVE = 0).

- **CPAS: RA Compare**

0: No effect.

1: Disables the RA Compare Interrupt (if TC_CMRx.WAVE = 1).

- **CPBS: RB Compare**

0: No effect.

1: Disables the RB Compare Interrupt (if TC_CMRx.WAVE = 1).

- **CPCS: RC Compare**

0: No effect.

1: Disables the RC Compare Interrupt.

- **LDRAS: RA Loading**

0: No effect.

1: Disables the RA Load Interrupt (if TC_CMRx.WAVE = 0).

- **LDRBS: RB Loading**

0: No effect.

1: Disables the RB Load Interrupt (if TC_CMRx.WAVE = 0).

Downloaded from Arrow.com.

- **ETRGS: External Trigger**

0: No effect.

1: Disables the External Trigger Interrupt.

Downloaded from Arrow.com.

48.7.13 TC Interrupt Mask Register

Name: TC_IMRx [x=0..2]

Address: 0x4000C02C (0)[0], 0x4000C06C (0)[1], 0x4000C0AC (0)[2], 0x4001002C (1)[0], 0x4001006C (1)[1], 0x400100AC (1)[2], 0x4001402C (2)[0], 0x4001406C (2)[1], 0x400140AC (2)[2], 0x4005402C (3)[0], 0x4005406C (3)[1], 0x400540AC (3)[2]

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: Counter Overflow**

0: The Counter Overflow Interrupt is disabled.

1: The Counter Overflow Interrupt is enabled.

- **LOVRS: Load Overrun**

0: The Load Overrun Interrupt is disabled.

1: The Load Overrun Interrupt is enabled.

- **CPAS: RA Compare**

0: The RA Compare Interrupt is disabled.

1: The RA Compare Interrupt is enabled.

- **CPBS: RB Compare**

0: The RB Compare Interrupt is disabled.

1: The RB Compare Interrupt is enabled.

- **CPCS: RC Compare**

0: The RC Compare Interrupt is disabled.

1: The RC Compare Interrupt is enabled.

- **LDRAS: RA Loading**

0: The Load RA Interrupt is disabled.

1: The Load RA Interrupt is enabled.

- **LDRBS: RB Loading**

0: The Load RB Interrupt is disabled.

1: The Load RB Interrupt is enabled.

Downloaded from Arrow.com.

- **ETRGS: External Trigger**

0: The External Trigger Interrupt is disabled.

1: The External Trigger Interrupt is enabled.

Downloaded from Arrow.com.

48.7.14 TC Extended Mode Register

Name: TC_EMRx [x=0..2]

Address: 0x4000C030 (0)[0], 0x4000C070 (0)[1], 0x4000C0B0 (0)[2], 0x40010030 (1)[0], 0x40010070 (1)[1], 0x400100B0 (1)[2], 0x40014030 (2)[0], 0x40014070 (2)[1], 0x400140B0 (2)[2], 0x40054030 (3)[0], 0x40054070 (3)[1], 0x400540B0 (3)[2]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	NODIVCLK
7	6	5	4	3	2	1	0
–	–	TRIGSRCB		–	–	TRIGSRCA	

- **TRIGSRCA: Trigger Source for Input A**

Value	Name	Description
0	EXTERNAL_TIOAx	The trigger/capture input A is driven by external pin TIOAx
1	PWMx	The trigger/capture input A is driven internally by PWMx

- **TRIGSRCB: Trigger Source for Input B**

Value	Name	Description
0	EXTERNAL_TIOBx	The trigger/capture input B is driven by external pin TIOBx
1	PWMx	For TC0 to TC10: The trigger/capture input B is driven internally by the comparator output (see Figure 48-16) of the PWMx. For TC11: The trigger/capture input B is driven internally by the GTSUCOMP signal of the Ethernet MAC (GMAC).

- **NODIVCLK: No Divided Clock**

0: The selected clock is defined by field TCCLKS in TC_CMRx.

1: The selected clock is peripheral clock and TCCLKS field (TC_CMRx) has no effect.

Downloaded from Arrow.com.

48.7.15 TC Block Control Register

Name: TC_BCR

Address: 0x4000C0C0 (0), 0x400100C0 (1), 0x400140C0 (2), 0x400540C0 (3)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	SYNC

- **SYNC: Synchro Command**

0: No effect.

1: Asserts the SYNC signal which generates a software trigger simultaneously for each of the channels.

Downloaded from Arrow.com.

48.7.16 TC Block Mode Register

Name: TC_BMR

Address: 0x4000C0C4 (0), 0x400100C4 (1), 0x400140C4 (2), 0x400540C4 (3)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	MAXFILT	

23	22	21	20	19	18	17	16
MAXFILT				–	–	IDXPHB	SWAP

15	14	13	12	11	10	9	8
INVIDX	INVB	INVA	EDGPHA	QDTRANS	SPEEDEN	POSEN	QDEN

7	6	5	4	3	2	1	0
–	–	TC2XC2S		TC1XC1S		TC0XC0S	

This register can only be written if the WPEN bit is cleared in the TC Write Protection Mode Register.

- **TC0XC0S: External Clock Signal 0 Selection**

Value	Name	Description
0	TCLK0	Signal connected to XC0: TCLK0
1	–	Reserved
2	TIOA1	Signal connected to XC0: TIOA1
3	TIOA2	Signal connected to XC0: TIOA2

- **TC1XC1S: External Clock Signal 1 Selection**

Value	Name	Description
0	TCLK1	Signal connected to XC1: TCLK1
1	–	Reserved
2	TIOA0	Signal connected to XC1: TIOA0
3	TIOA2	Signal connected to XC1: TIOA2

- **TC2XC2S: External Clock Signal 2 Selection**

Value	Name	Description
0	TCLK2	Signal connected to XC2: TCLK2
1	–	Reserved
2	TIOA0	Signal connected to XC2: TIOA0
3	TIOA1	Signal connected to XC2: TIOA1

Downloaded from Arrow.com.

- **QDEN: Quadrature Decoder Enabled**

0: Disabled.

1: Enables the QDEC (filter, edge detection and quadrature decoding).

Quadrature decoding (direction change) can be disabled using QDTRANS bit.

One of the POSEN or SPEEDEN bits must be also enabled.

- **POSEN: Position Enabled**

0: Disable position.

1: Enables the position measure on channel 0 and 1.

- **SPEEDEN: Speed Enabled**

0: Disabled.

1: Enables the speed measure on channel 0, the time base being provided by channel 2.

- **QDTRANS: Quadrature Decoding Transparent**

0: Full quadrature decoding logic is active (direction change detected).

1: Quadrature decoding logic is inactive (direction change inactive) but input filtering and edge detection are performed.

- **EDGPHA: Edge on PHA Count Mode**

0: Edges are detected on PHA only.

1: Edges are detected on both PHA and PHB.

- **INVA: Inverted PHA**

0: PHA (TIOA0) is directly driving the QDEC.

1: PHA is inverted before driving the QDEC.

- **INVB: Inverted PHB**

0: PHB (TIOB0) is directly driving the QDEC.

1: PHB is inverted before driving the QDEC.

- **INVIDX: Inverted Index**

0: IDX (TIOA1) is directly driving the QDEC.

1: IDX is inverted before driving the QDEC.

- **SWAP: Swap PHA and PHB**

0: No swap between PHA and PHB.

1: Swap PHA and PHB internally, prior to driving the QDEC.

- **IDXPHB: Index Pin is PHB Pin**

0: IDX pin of the rotary sensor must drive TIOA1.

1: IDX pin of the rotary sensor must drive TIOB0.

- **MAXFILT: Maximum Filter**

1–63: Defines the filtering capabilities.

Pulses with a period shorter than MAXFILT+1 peripheral clock cycles are discarded.

Downloaded from Arrow.com.

48.7.17 TC QDEC Interrupt Enable Register

Name: TC_QIER

Address: 0x4000C0C8 (0), 0x400100C8 (1), 0x400140C8 (2), 0x400540C8 (3)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	QERR	DIRCHG	IDX

- **IDX: Index**

0: No effect.

1: Enables the interrupt when a rising edge occurs on IDX input.

- **DIRCHG: Direction Change**

0: No effect.

1: Enables the interrupt when a change on rotation direction is detected.

- **QERR: Quadrature Error**

0: No effect.

1: Enables the interrupt when a quadrature error occurs on PHA, PHB.

Downloaded from Arrow.com.

48.7.18 TC QDEC Interrupt Disable Register

Name: TC_QIDR

Address: 0x4000C0CC (0), 0x400100CC (1), 0x400140CC (2), 0x400540CC (3)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	QERR	DIRCHG	IDX

- **IDX: Index**

0: No effect.

1: Disables the interrupt when a rising edge occurs on IDX input.

- **DIRCHG: Direction Change**

0: No effect.

1: Disables the interrupt when a change on rotation direction is detected.

- **QERR: Quadrature Error**

0: No effect.

1: Disables the interrupt when a quadrature error occurs on PHA, PHB.

Downloaded from Arrow.com.

48.7.19 TC QDEC Interrupt Mask Register

Name: TC_QIMR

Address: 0x4000C0D0 (0), 0x400100D0 (1), 0x400140D0 (2), 0x400540D0 (3)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	QERR	DIRCHG	IDX

- **IDX: Index**

0: The interrupt on IDX input is disabled.

1: The interrupt on IDX input is enabled.

- **DIRCHG: Direction Change**

0: The interrupt on rotation direction change is disabled.

1: The interrupt on rotation direction change is enabled.

- **QERR: Quadrature Error**

0: The interrupt on quadrature error is disabled.

1: The interrupt on quadrature error is enabled.

Downloaded from Arrow.com.

48.7.20 TC QDEC Interrupt Status Register

Name: TC_QISR

Address: 0x4000C0D4 (0), 0x400100D4 (1), 0x400140D4 (2), 0x400540D4 (3)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	DIR

7	6	5	4	3	2	1	0
–	–	–	–	–	QERR	DIRCHG	IDX

- **IDX: Index**

0: No Index input change since the last read of TC_QISR.

1: The IDX input has changed since the last read of TC_QISR.

- **DIRCHG: Direction Change**

0: No change on rotation direction since the last read of TC_QISR.

1: The rotation direction changed since the last read of TC_QISR.

- **QERR: Quadrature Error**

0: No quadrature error since the last read of TC_QISR.

1: A quadrature error occurred since the last read of TC_QISR.

- **DIR: Direction**

Returns an image of the actual rotation direction.

Downloaded from Arrow.com.

48.7.21 TC Fault Mode Register

Name: TC_FMR

Address: 0x4000C0D8 (0), 0x400100D8 (1), 0x400140D8 (2), 0x400540D8 (3)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	ENCF1	ENCF0

This register can only be written if the WPEN bit is cleared in the TC Write Protection Mode Register.

- **ENCF0: Enable Compare Fault Channel 0**

0: Disables the FAULT output source (CPCS flag) from channel 0.

1: Enables the FAULT output source (CPCS flag) from channel 0.

- **ENCF1: Enable Compare Fault Channel 1**

0: Disables the FAULT output source (CPCS flag) from channel 1.

1: Enables the FAULT output source (CPCS flag) from channel 1.

Downloaded from Arrow.com.

48.7.22 TC Write Protection Mode Register

Name: TC_WPMR

Address: 0x4000C0E4 (0), 0x400100E4 (1), 0x400140E4 (2), 0x400540E4 (3)

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							

23	22	21	20	19	18	17	16
WPKEY							

15	14	13	12	11	10	9	8
WPKEY							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x54494D (“TIM” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x54494D (“TIM” in ASCII).

The Timer Counter clock of the first channel must be enabled to access this register.

See Section 48.6.19 “Register Write Protection” for a list of registers that can be write-protected and Timer Counter clock conditions.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x54494D	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

Downloaded from Arrow.com.

49. Pulse Width Modulation Controller (PWM)

49.1 Description

The Pulse Width Modulation Controller (PWM) generates output pulses on 4 channels independently according to parameters defined per channel. Each channel controls two complementary square output waveforms. Characteristics of the output waveforms such as period, duty-cycle, polarity and dead-times (also called dead-bands or non-overlapping times) are configured through the user interface. Each channel selects and uses one of the clocks provided by the clock generator. The clock generator provides several clocks resulting from the division of the PWM peripheral clock. External triggers can be managed to allow output pulses to be modified in real time.

All accesses to the PWM are made through registers mapped on the peripheral bus. All channels integrate a double buffering system in order to prevent an unexpected output waveform while modifying the period, the spread spectrum, the duty-cycle or the dead-times.

Channels can be linked together as synchronous channels to be able to update their duty-cycle or dead-times at the same time.

The update of duty-cycles of synchronous channels can be performed by the DMA Controller channel which offers buffer transfer without processor Intervention.

The PWM includes a spread-spectrum counter to allow a constantly varying period (only for Channel 0). This counter may be useful to minimize electromagnetic interference or to reduce the acoustic noise of a PWM driven motor.

The PWM provides 8 independent comparison units capable of comparing a programmed value to the counter of the synchronous channels (counter of channel 0). These comparisons are intended to generate software interrupts, to trigger pulses on the 2 independent event lines (in order to synchronize ADC conversions with a lot of flexibility independently of the PWM outputs) and to trigger DMA Controller transfer requests.

PWM outputs can be overridden synchronously or asynchronously to their channel counter.

The PWM provides a fault protection mechanism with 8 fault inputs, capable to detect a fault condition and to override the PWM outputs asynchronously (outputs forced to ‘0’, ‘1’ or Hi-Z).

For safety usage, some configuration registers are write-protected.

Downloaded from Arrow.com.

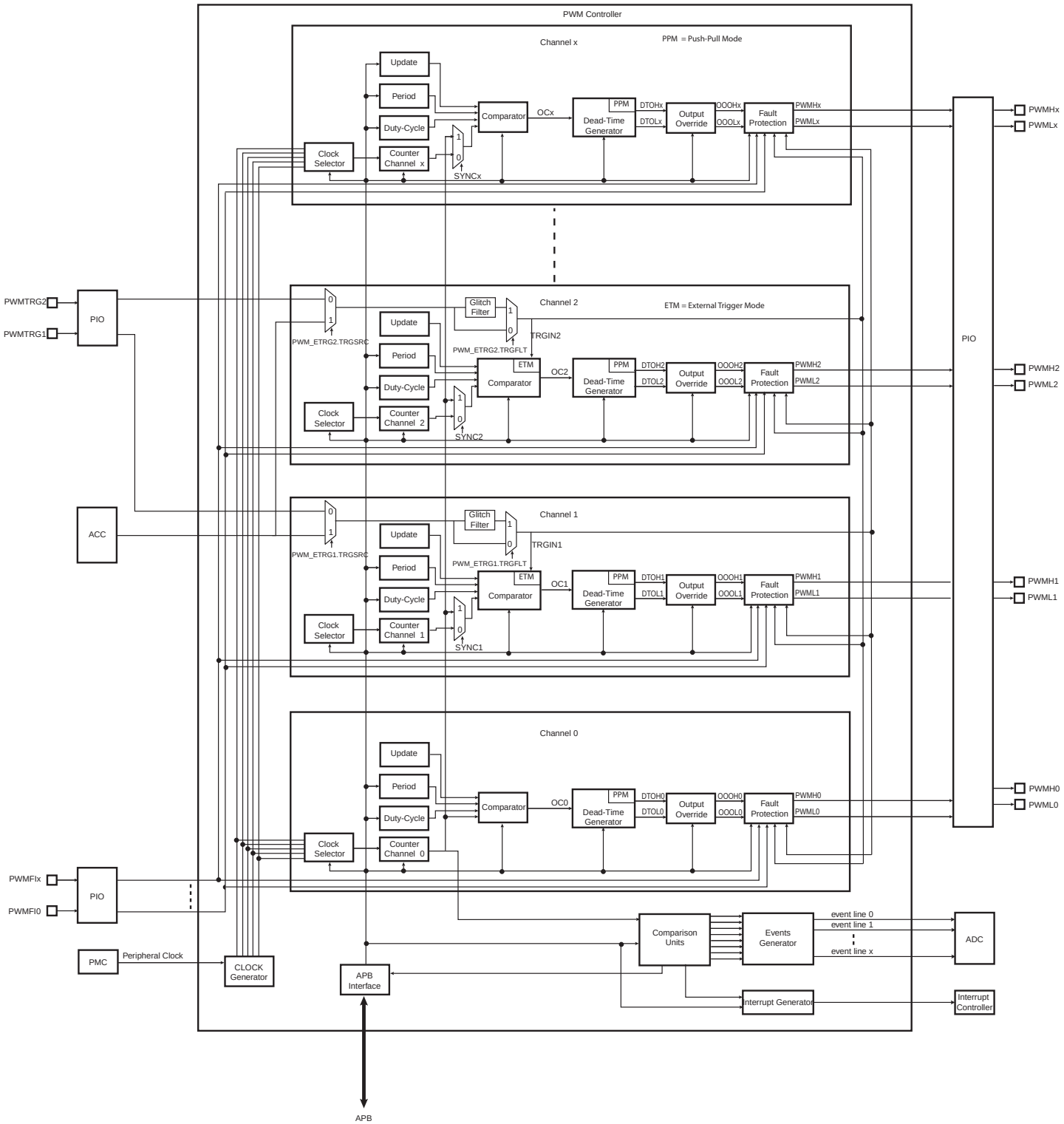
49.2 Embedded Characteristics

- 4 Channels
- Common Clock Generator Providing Thirteen Different Clocks
  - A Modulo n Counter Providing Eleven Clocks
  - Two Independent Linear Dividers Working on Modulo n Counter Outputs
- Independent Channels
  - Independent 16-bit Counter for Each Channel
  - Independent Complementary Outputs with 12-bit Dead-Time Generator (Also Called Dead-Band or Non-Overlapping Time) for Each Channel
  - Independent Push-Pull Mode for Each Channel
  - Independent Enable Disable Command for Each Channel
  - Independent Clock Selection for Each Channel
  - Independent Period, Duty-Cycle and Dead-Time for Each Channel
  - Independent Double Buffering of Period, Duty-Cycle and Dead-Times for Each Channel
  - Independent Programmable Selection of The Output Waveform Polarity for Each Channel, with Double Buffering
  - Independent Programmable Center- or Left-aligned Output Waveform for Each Channel
  - Independent Output Override for Each Channel
  - Independent Interrupt for Each Channel, at Each Period for Left-Aligned or Center-Aligned Configuration
  - Independent Update Time Selection of Double Buffering Registers (Polarity, Duty Cycle) for Each Channel, at Each Period for Left-Aligned or Center-Aligned Configuration
- External Trigger Input Management (e.g., for DC/DC or Lighting Control)
  - External PWM Reset Mode
  - External PWM Start Mode
  - Cycle-By-Cycle Duty Cycle Mode
  - Leading-Edge Blanking
- 2 2-bit Gray Up/Down Channels for Stepper Motor Control
- Spread Spectrum Counter to Allow a Constantly Varying Duty Cycle (only for Channel 0)
- Synchronous Channel Mode
  - Synchronous Channels Share the Same Counter
  - Mode to Update the Synchronous Channels Registers after a Programmable Number of Periods
  - Synchronous Channels Supports Connection of one DMA Controller Channel Which Offers Buffer Transfer Without Processor Intervention To Update Duty-Cycle Registers
- 2 Independent Events Lines Intended to Synchronize ADC Conversions
  - Programmable delay for Events Lines to delay ADC measurements
- 8 Comparison Units Intended to Generate Interrupts, Pulses on Event Lines and DMA Controller Transfer Requests
- 8 Programmable Fault Inputs Providing an Asynchronous Protection of PWM Outputs
  - 3 User Driven through PIO Inputs
  - PMC Driven when Crystal Oscillator Clock Fails
  - ADC Controller Driven through Configurable Comparison Function
  - Analog Comparator Controller Driven
  - Timer/Counter Driven through Configurable Comparison Function
- Register Write Protection

Downloaded from Arrow.com.

49.3 Block Diagram

Figure 49-1. Pulse Width Modulation Controller Block Diagram



Downloaded from Arrow.com.

49.4 I/O Lines Description

Each channel outputs two complementary external I/O lines.

Table 49-1. I/O Line Description

Name	Description	Type
PWMHx	PWM Waveform Output High for channel x	Output
PWMLx	PWM Waveform Output Low for channel x	Output
PWMFIx	PWM Fault Input x	Input
PWMTRGy	PWM Trigger Input y	Input

49.5 Product Dependencies

49.5.1 I/O Lines

The pins used for interfacing the PWM are multiplexed with PIO lines. The programmer must first program the PIO controller to assign the desired PWM pins to their peripheral function. If I/O lines of the PWM are not used by the application, they can be used for other purposes by the PIO controller.

All of the PWM outputs may or may not be enabled. If an application requires only four channels, then only four PIO lines are assigned to PWM outputs.

Table 49-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
PWM0	PWMC0_PWMEXTRG0	PA10	B
PWM0	PWMC0_PWMEXTRG1	PA22	B
PWM0	PWMC0_PWMFI0	PA9	C
PWM0	PWMC0_PWMFI1	PD8	B
PWM0	PWMC0_PWMFI2	PD9	B
PWM0	PWMC0_PWMH0	PA0	A
PWM0	PWMC0_PWMH0	PA11	B
PWM0	PWMC0_PWMH0	PA23	B
PWM0	PWMC0_PWMH0	PB0	A
PWM0	PWMC0_PWMH0	PD11	B
PWM0	PWMC0_PWMH0	PD20	A
PWM0	PWMC0_PWMH1	PA2	A
PWM0	PWMC0_PWMH1	PA12	B
PWM0	PWMC0_PWMH1	PA24	B
PWM0	PWMC0_PWMH1	PB1	A
PWM0	PWMC0_PWMH1	PD21	A
PWM0	PWMC0_PWMH2	PA13	B
PWM0	PWMC0_PWMH2	PA25	B
PWM0	PWMC0_PWMH2	PB4	B
PWM0	PWMC0_PWMH2	PC19	B
PWM0	PWMC0_PWMH2	PD22	A

Downloaded from Arrow.com.

Table 49-2. I/O Lines

PWM0	PWMC0_PWMH3	PA7	B
PWM0	PWMC0_PWMH3	PA14	B
PWM0	PWMC0_PWMH3	PA17	C
PWM0	PWMC0_PWMH3	PC13	B
PWM0	PWMC0_PWMH3	PC21	B
PWM0	PWMC0_PWMH3	PD23	A
PWM0	PWMC0_PWML0	PA1	A
PWM0	PWMC0_PWML0	PA19	B
PWM0	PWMC0_PWML0	PB5	B
PWM0	PWMC0_PWML0	PC0	B
PWM0	PWMC0_PWML0	PD10	B
PWM0	PWMC0_PWML0	PD24	A
PWM0	PWMC0_PWML1	PA20	B
PWM0	PWMC0_PWML1	PB12	A
PWM0	PWMC0_PWML1	PC1	B
PWM0	PWMC0_PWML1	PC18	B
PWM0	PWMC0_PWML1	PD25	A
PWM0	PWMC0_PWML2	PA16	C
PWM0	PWMC0_PWML2	PA30	A
PWM0	PWMC0_PWML2	PB13	A
PWM0	PWMC0_PWML2	PC2	B
PWM0	PWMC0_PWML2	PC20	B
PWM0	PWMC0_PWML2	PD26	A
PWM0	PWMC0_PWML3	PA15	C
PWM0	PWMC0_PWML3	PC3	B
PWM0	PWMC0_PWML3	PC15	B
PWM0	PWMC0_PWML3	PC22	B
PWM0	PWMC0_PWML3	PD27	A
PWM1	PWMC1_PWMEXTRG0	PA30	B
PWM1	PWMC1_PWMEXTRG1	PA18	A
PWM1	PWMC1_PWMFI0	PA21	C
PWM1	PWMC1_PWMFI1	PA26	D
PWM1	PWMC1_PWMFI2	PA28	D
PWM1	PWMC1_PWMH0	PA12	C
PWM1	PWMC1_PWMH0	PD1	B
PWM1	PWMC1_PWMH1	PA14	C
PWM1	PWMC1_PWMH1	PD3	B
PWM1	PWMC1_PWMH2	PA31	D

Downloaded from Arrow.com.

Table 49-2. I/O Lines

PWM1	PWMC1_PWMH2	PD5	B
PWM1	PWMC1_PWMH3	PA8	A
PWM1	PWMC1_PWMH3	PD7	B
PWM1	PWMC1_PWML0	PA11	C
PWM1	PWMC1_PWML0	PD0	B
PWM1	PWMC1_PWML1	PA13	C
PWM1	PWMC1_PWML1	PD2	B
PWM1	PWMC1_PWML2	PA23	D
PWM1	PWMC1_PWML2	PD4	B
PWM1	PWMC1_PWML3	PA5	A
PWM1	PWMC1_PWML3	PD6	B

49.5.2 Power Management

The PWM is not continuously clocked. The programmer must first enable the PWM clock in the Power Management Controller (PMC) before using the PWM. However, if the application does not require PWM operations, the PWM clock can be stopped when not needed and be restarted later. In this case, the PWM will resume its operations where it left off.

49.5.3 Interrupt Sources

The PWM interrupt line is connected on one of the internal sources of the Interrupt Controller. Using the PWM interrupt requires the Interrupt Controller to be programmed first.

Table 49-3. Peripheral IDs

Instance	ID
PWM0	31
PWM1	60

49.5.4 Fault Inputs

The PWM has the fault inputs connected to the different modules. Refer to the implementation of these modules within the product for detailed information about the fault generation procedure. The PWM receives faults from PIO inputs, the PMC, the ADC controller, the Analog Comparator Controller and Timer/Counters.

Table 49-4. Fault Inputs

Fault Generator	External PWM Fault Input Number	Polarity Level(1)	Fault Input ID
PWM0			
PA9	PWMC0_PWMFI0	User-defined	0
PD8	PWMC0_PWMFI1	User-defined	1
PD9	PWMC0_PWMFI2	User-defined	2
Main OSC (PMC)	–	To be configured to 1	3
AFEC0	–	To be configured to 1	4
AFEC1	–	To be configured to 1	5
ACC	–	To be configured to 1	6

Downloaded from Arrow.com.

Table 49-4. Fault Inputs (Continued)

Fault Generator	External PWM Fault Input Number	Polarity Level[1]	Fault Input ID
Timer0	–	To be configured to 1	7
PWM1			
PA21	PWMC1_PWMFI0	User-defined	0
PA26	PWMC1_PWMFI1	User-defined	1
PA28	PWMC1_PWMFI2	User-defined	2
Main OSC (PMC)	–	To be configured to 1	3
AFEC0	–	To be configured to 1	4
AFEC1	–	To be configured to 1	5
ACC	–	To be configured to 1	6
Timer1	–	To be configured to 1	7

Note: 1. FPOL field in PWMC_FMR.

Downloaded from Arrow.com.

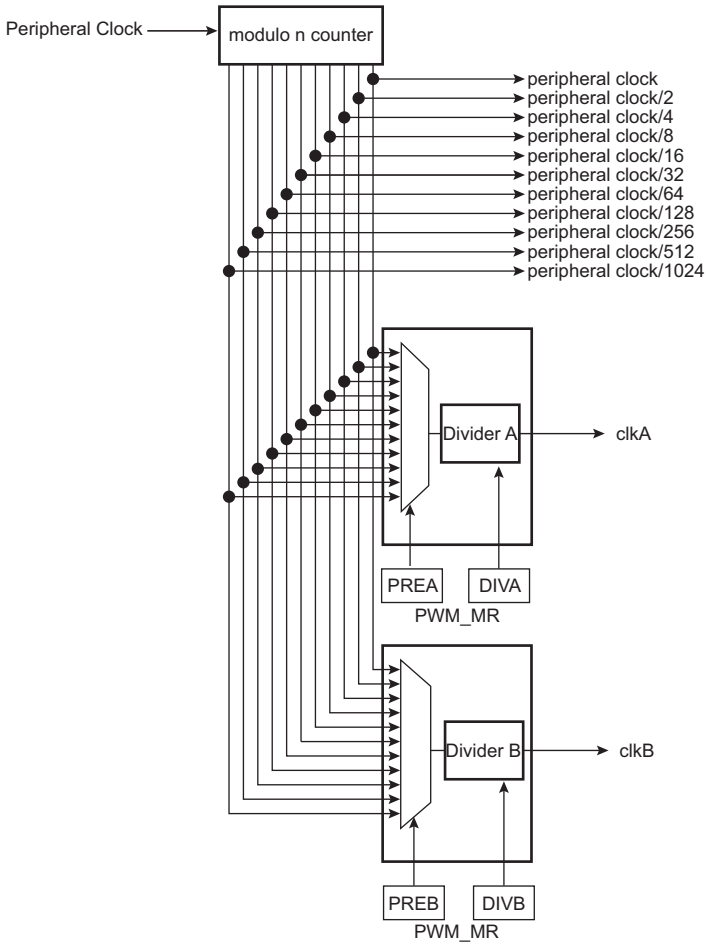
49.6 Functional Description

The PWM controller is primarily composed of a clock generator module and 4 channels.

- Clocked by the peripheral clock, the clock generator module provides 13 clocks.
- Each channel can independently choose one of the clock generator outputs.
- Each channel generates an output waveform with attributes that can be defined independently for each channel through the user interface registers.

49.6.1 PWM Clock Generator

Figure 49-2. Functional View of the Clock Generator Block Diagram



The PWM peripheral clock is divided in the clock generator module to provide different clocks available for all channels. Each channel can independently select one of the divided clocks.

The clock generator is divided into different blocks:

- a modulo n counter which provides 11 clocks: $f_{peripheral\ clock}$, $f_{peripheral\ clock}/2$, $f_{peripheral\ clock}/4$, $f_{peripheral\ clock}/8$, $f_{peripheral\ clock}/16$, $f_{peripheral\ clock}/32$, $f_{peripheral\ clock}/64$, $f_{peripheral\ clock}/128$, $f_{peripheral\ clock}/256$, $f_{peripheral\ clock}/512$, $f_{peripheral\ clock}/1024$
- two linear dividers (1, 1/2, 1/3, ... 1/255) that provide two separate clocks: clkA and clkB

Each linear divider can independently divide one of the clocks of the modulo n counter. The selection of the clock to be divided is made according to the PREA (PREB) field of the PWM Clock register (PWM_CLK). The resulting clock clkA (clkB) is the clock selected divided by DIVA (DIVB) field value.

Downloaded from Arrow.com.

After a reset of the PWM controller, DIVA (DIVB) and PREA (PREB) are set to ‘0’. This implies that after reset clkA (clkB) are turned off.

At reset, all clocks provided by the modulo n counter are turned off except the peripheral clock. This situation is also true when the PWM peripheral clock is turned off through the Power Management Controller.

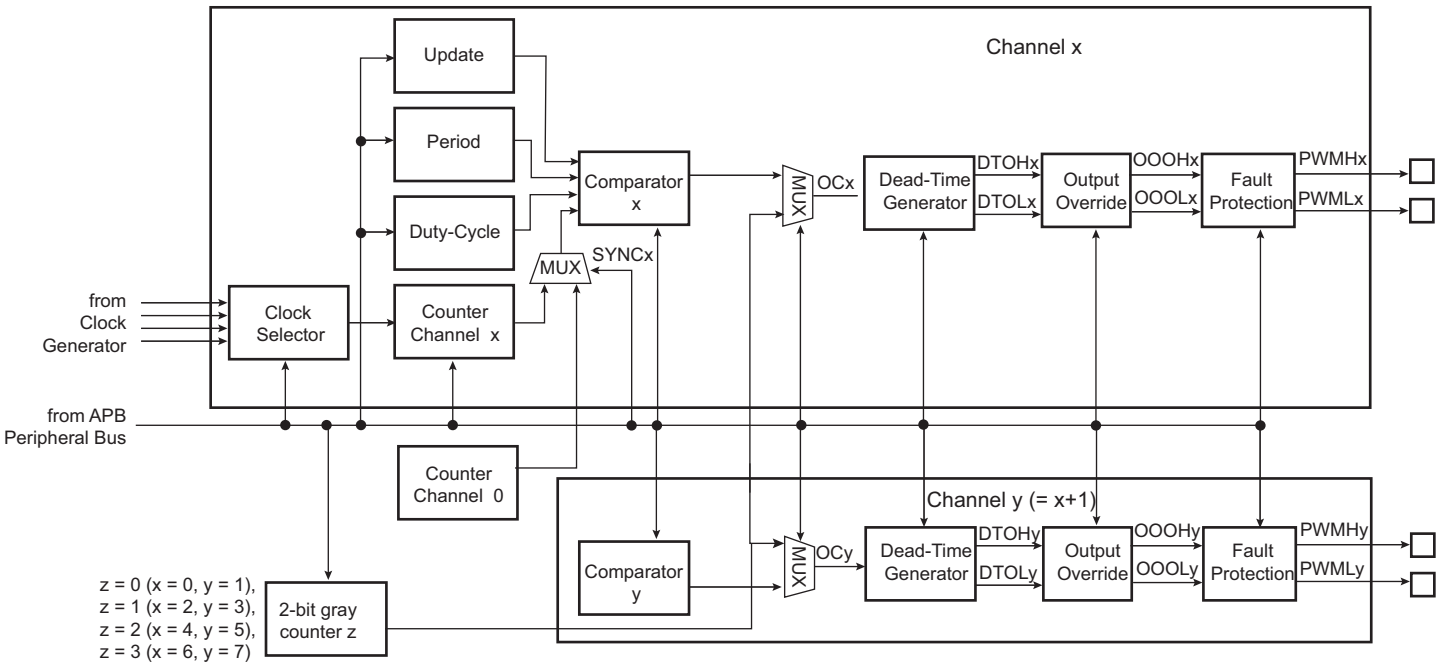
CAUTION:

Before using the PWM controller, the programmer must first enable the peripheral clock in the Power Management Controller (PMC).

49.6.2 PWM Channel

49.6.2.1 Channel Block Diagram

Figure 49-3. Functional View of the Channel Block Diagram



Each of the 4 channels is composed of six blocks:

- A clock selector which selects one of the clocks provided by the clock generator (described in Section 49.6.1 “PWM Clock Generator”).
- A counter clocked by the output of the clock selector. This counter is incremented or decremented according to the channel configuration and comparators matches. The size of the counter is 16 bits.
- A comparator used to compute the OCx output waveform according to the counter value and the configuration. The counter value can be the one of the channel counter or the one of the channel 0 counter according to SYNCx bit in the PWM Sync Channels Mode Register (PWM_SCM).
- A 2-bit configurable gray counter enables the stepper motor driver. One gray counter drives 2 channels.
- A dead-time generator providing two complementary outputs (DTOHx/DTOLx) which allows to drive external power control switches safely.
- An output override block that can force the two complementary outputs to a programmed value (OOOHx/OOOLx).
- An asynchronous fault protection mechanism that has the highest priority to override the two complementary outputs (PWMHx/PWMLx) in case of fault detection (outputs forced to ‘0’, ‘1’ or Hi-Z).

Downloaded from Arrow.com.

49.6.2.2 Comparator

The comparator continuously compares its counter value with the channel period defined by CPRD in the PWM Channel Period Register (PWM_CPRDx) and the duty-cycle defined by CDTY in the PWM Channel Duty Cycle Register (PWM_CDTYx) to generate an output signal OCx accordingly.

The different properties of the waveform of the output OCx are:

- the ***clock selection***. The channel counter is clocked by one of the clocks provided by the clock generator described in the previous section. This channel parameter is defined in the CPRE field of the PWM Channel Mode Register (PWM_CMRx). This field is reset at '0'.
- the ***waveform period***. This channel parameter is defined in the CPRD field of the PWM_CPRDx register.
  If the waveform is left-aligned, then the output waveform period depends on the counter source clock and can be calculated:
  By using the PWM peripheral clock divided by a given prescaler value "X" (where $X = 2^{PREA}$ is 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula is:

  $$\frac{(X \times CPRD)}{f_{peripheral\ clock}}$$

  By using the PWM peripheral clock divided by a given prescaler value "X" (see above) and by either the DIVA or the DIVB divider. The formula becomes, respectively:

  $$\frac{(X \times CRPD \times DIVA)}{f_{peripheral\ clock}} \ or \ \frac{(X \times CRPD \times DIVB)}{f_{peripheral\ clock}}$$

  If the waveform is center-aligned, then the output waveform period depends on the counter source clock and can be calculated:
  By using the PWM peripheral clock divided by a given prescaler value "X" (where $X = 2^{PREA}$ is 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula is:

  $$\frac{(2 \times X \times CPRD)}{f_{peripheral\ clock}}$$

  By using the PWM peripheral clock divided by a given prescaler value "X" (see above) and by either the DIVA or the DIVB divider. The formula becomes, respectively:

  $$\frac{(2 \times X \times CPRD \times DIVA)}{f_{peripheral\ clock}} \ or$$
  $$\frac{(2 \times X \times CPRD \times DIVB)}{f_{peripheral\ clock}}$$

- the ***waveform duty-cycle***. This channel parameter is defined in the CDTY field of the PWM_CDTYx register.
  If the waveform is left-aligned, then:

  $$\text{duty cycle} = \frac{(period - 1/fchannel_x_clock \times CDTY)}{period}$$

  If the waveform is center-aligned, then:

  $$\text{duty cycle} = \frac{((period/2) - 1/fchannel_x_clock \times CDTY))}{(period/2)}$$

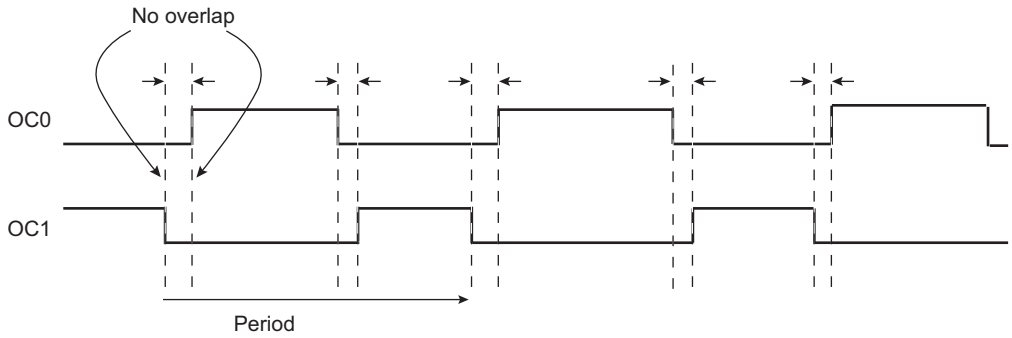
- the ***waveform polarity.*** At the beginning of the period, the signal can be at high or low level. This property is defined in the CPOL bit of PWM_CMRx. By default, the signal starts by a low level. The DPOLI bit in PWM_CMRx defines the PWM polarity when the channel is disabled (CHIDx = 0 in PWM_SR). For more details, see Figure 49-5.

Downloaded from Arrow.com.

- DPOLI = 0: PWM polarity when the channel is disabled is the same as the one defined for the beginning of the PWM period.
- DPOLI = 1: PWM polarity when the channel is disabled is inverted compared to the one defined for the beginning of the PWM period.

- the ***waveform alignment***. The output waveform can be left- or center-aligned. Center-aligned waveforms can be used to generate non-overlapped waveforms. This property is defined in the CALG bit of PWM_CMRx. The default mode is left-aligned.

Figure 49-4. Non-Overlapped Center-Aligned Waveforms



Note: 1. See Figure 49-5 for a detailed description of center-aligned waveforms.

When center-aligned, the channel counter increases up to CPRD and decreases down to 0. This ends the period.

When left-aligned, the channel counter increases up to CPRD and is reset. This ends the period.

Thus, for the same CPRD value, the period for a center-aligned channel is twice the period for a left-aligned channel.

Waveforms are fixed at 0 when:

- CDTY = CPRD and CPOL = 0 (Note that if TRGMODE = MODE3, the PWM waveform switches to 1 at the external trigger event (see Section 49.6.5.3 “Cycle-By-Cycle Duty Mode”)).
- CDTY = 0 and CPOL = 1

Waveforms are fixed at 1 (once the channel is enabled) when:

- CDTY = 0 and CPOL = 0
- CDTY = CPRD and CPOL = 1 (Note that if TRGMODE = MODE3, the PWM waveform switches to 0 at the external trigger event (see Section 49.6.5.3 “Cycle-By-Cycle Duty Mode”)).

The waveform polarity must be set before enabling the channel. This immediately affects the channel output level.

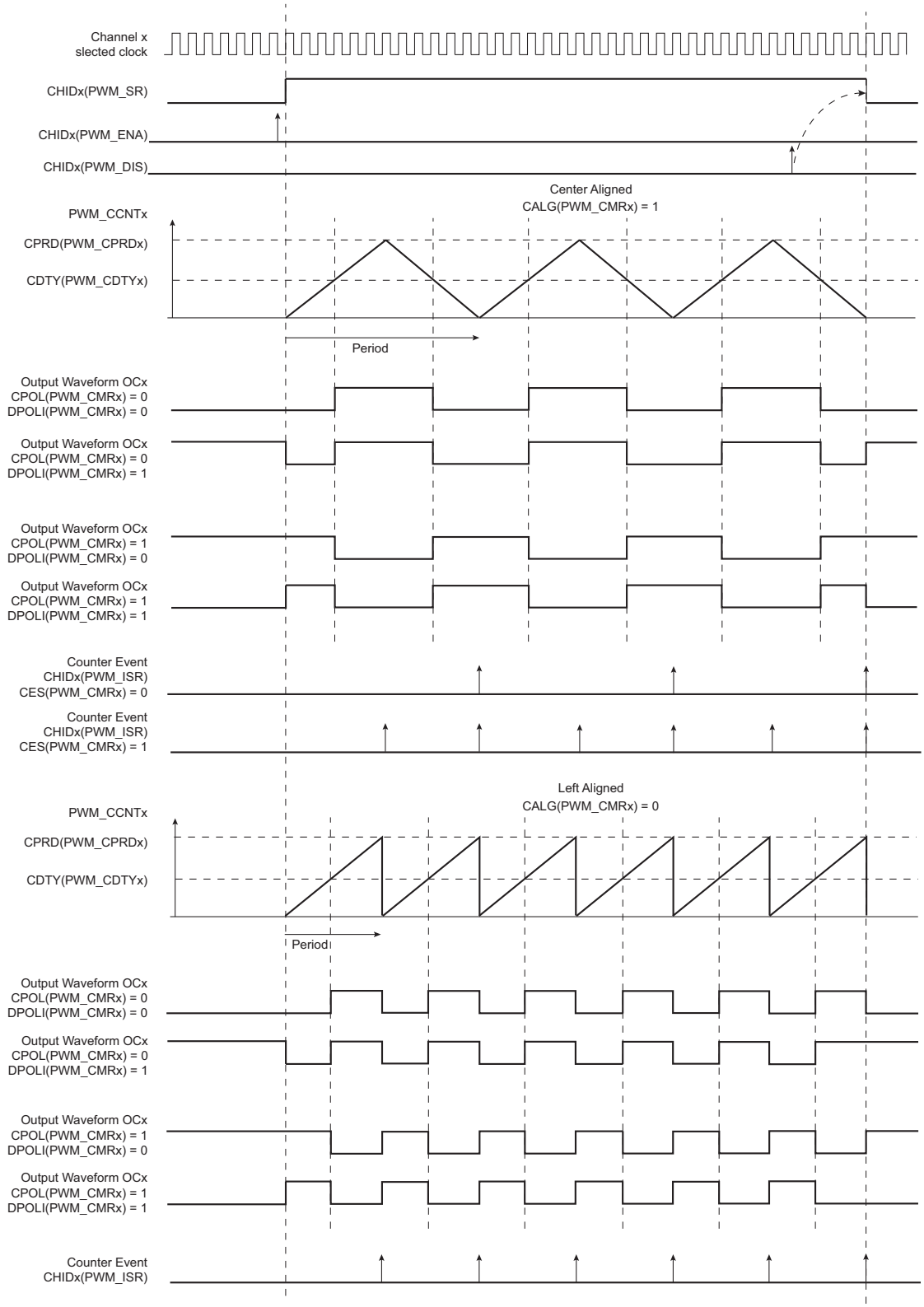
Modifying CPOL in PWM Channel Mode Register while the channel is enabled can lead to an unexpected behavior of the device being driven by PWM.

In addition to generating the output signals OCx, the comparator generates interrupts depending on the counter value. When the output waveform is left-aligned, the interrupt occurs at the end of the counter period. When the output waveform is center-aligned, the bit CES of PWM_CMRx defines when the channel counter interrupt occurs. If CES is set to ‘0’, the interrupt occurs at the end of the counter period. If CES is set to ‘1’, the interrupt occurs at the end of the counter period and at half of the counter period.

Figure 49-5 illustrates the counter interrupts depending on the configuration.

Downloaded from Arrow.com.

Figure 49-5. Waveform Properties



Downloaded from Arrow.com.

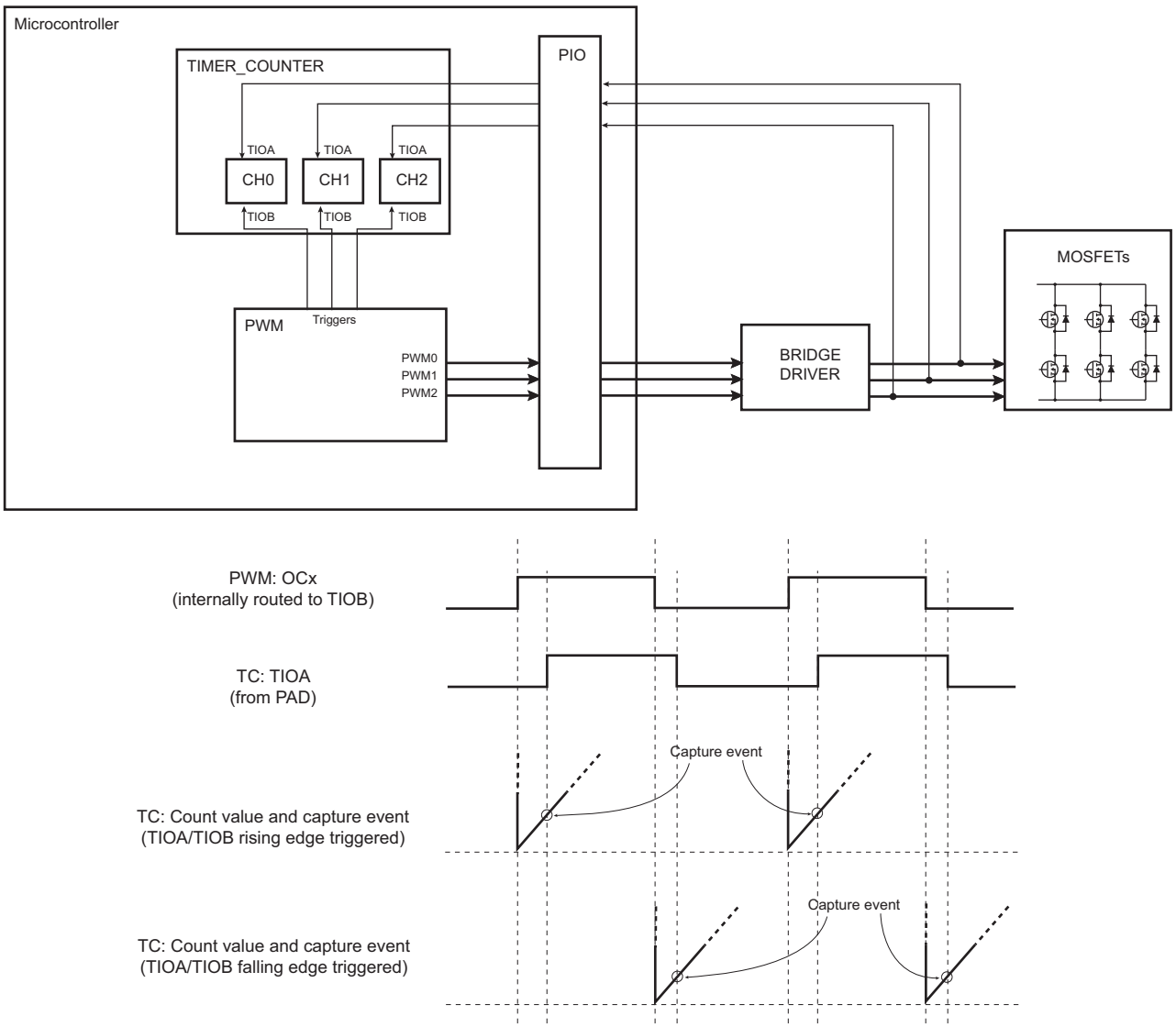
49.6.2.3 Trigger Selection for Timer Counter

The PWM controller can be used as a trigger source for the Timer Counter (TC) to achieve the two application examples described below.

Delay Measurement

To measure the delay between the channel x comparator output (OCx) and the feedback from the bridge driver of the MOSFETs (see Figure 49-6), the bit TCTS in the PWM Channel Mode Register must be at 0. This defines the comparator output of the channel x as the TC trigger source. The TIOB trigger (TC internal input) is used to start the TC; the TIOA input (from PAD) is used to capture the delay.

Figure 49-6. Triggering the TC: Delay Measurement

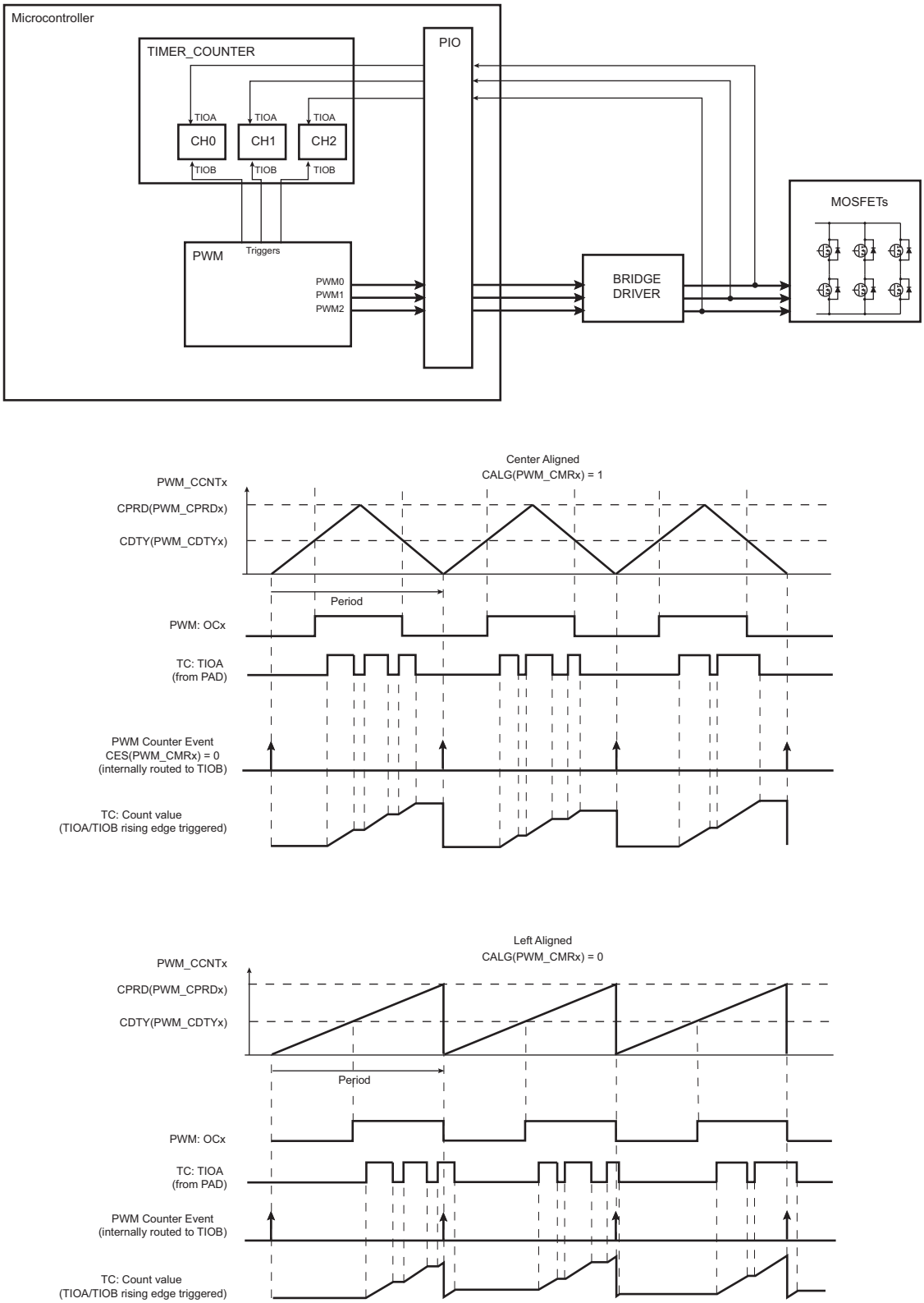


Downloaded from Arrow.com.

Cumulated ON Time Measurement

To measure the cumulated “ON” time of MOSFETs (see Figure 49-7), the bit TCTS of the PWM Channel Mode Register must be set to 1 to define the counter event (see Figure 49-5) as the Timer Counter trigger source.

Figure 49-7. Triggering the TC: Cumulated “ON” Time Measurement



Downloaded from Arrow.com.

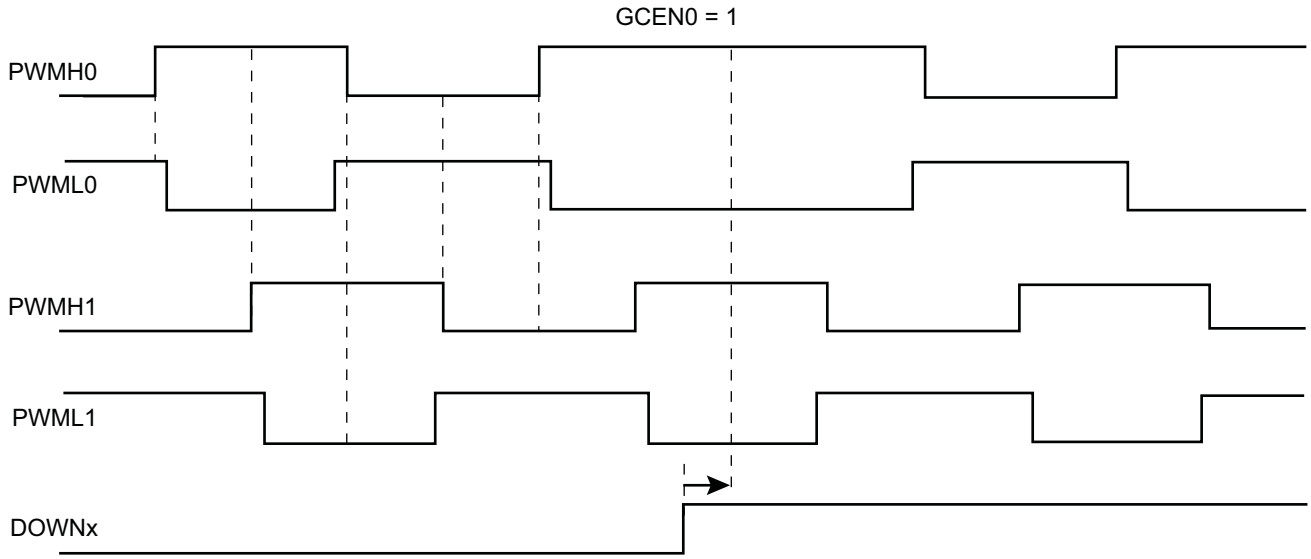
49.6.2.4 2-bit Gray Up/Down Counter for Stepper Motor

A pair of channels may provide a 2-bit gray count waveform on two outputs. Dead-time generator and other downstream logic can be configured on these channels.

Up or Down Count mode can be configured on-the-fly by means of PWM_SMMR configuration registers.

When GCEN0 is set to ‘1’, channels 0 and 1 outputs are driven with gray counter.

Figure 49-8. 2-bit Gray Up/Down Counter



49.6.2.5 Dead-Time Generator

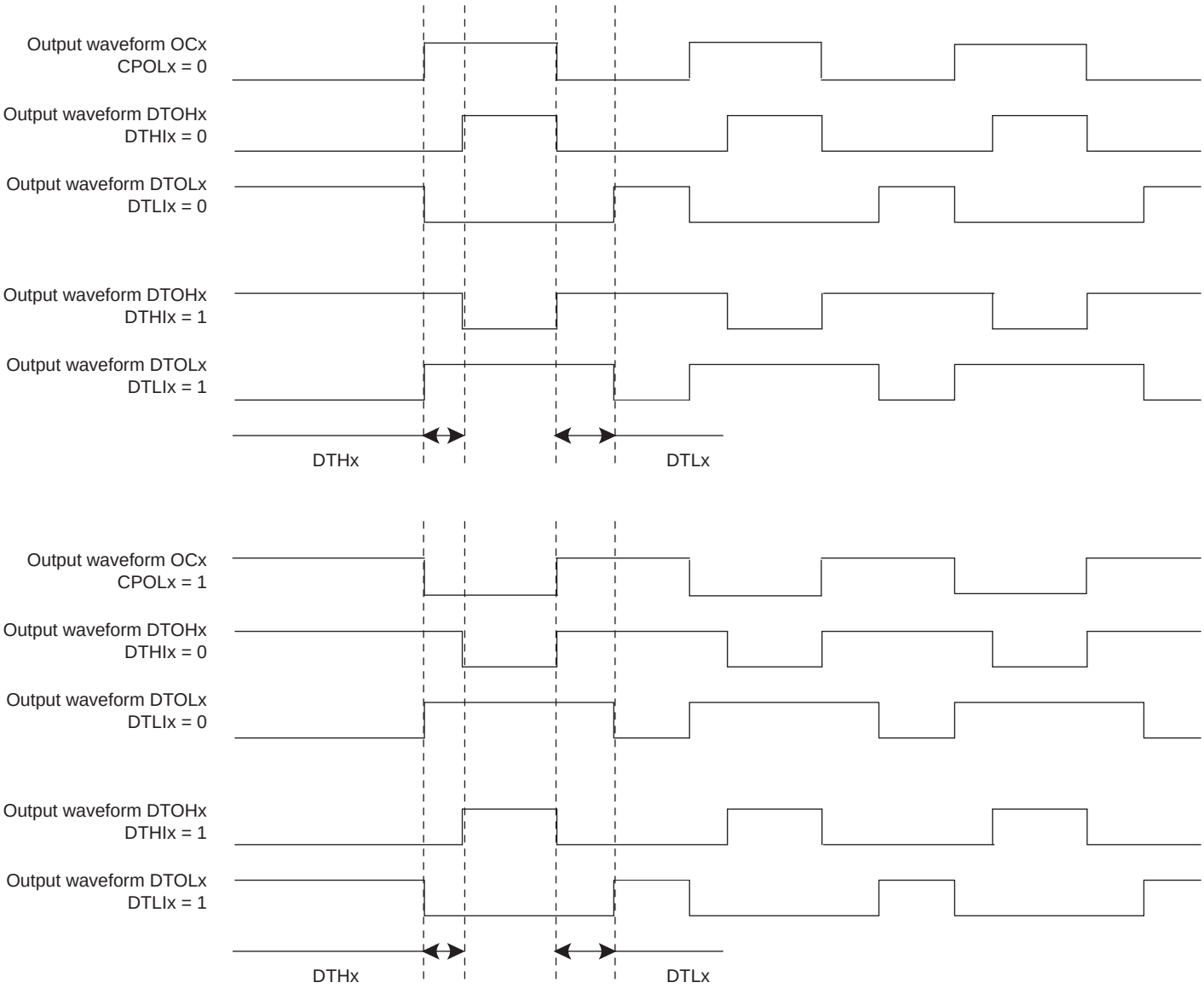
The dead-time generator uses the comparator output OCx to provide the two complementary outputs DTOHx and DTOLx, which allows the PWM macrocell to drive external power control switches safely. When the dead-time generator is enabled by setting the bit DTE to 1 or 0 in the PWM Channel Mode Register (PWM_CMRx), dead-times (also called dead-bands or non-overlapping times) are inserted between the edges of the two complementary outputs DTOHx and DTOLx. Note that enabling or disabling the dead-time generator is allowed only if the channel is disabled.

The dead-time is adjustable by the PWM Channel Dead Time Register (PWM_DTx). Each output of the dead-time generator can be adjusted separately by DTH and DTL. The dead-time values can be updated synchronously to the PWM period by using the PWM Channel Dead Time Update Register (PWM_DTUPDx).

The dead-time is based on a specific counter which uses the same selected clock that feeds the channel counter of the comparator. Depending on the edge and the configuration of the dead-time, DTOHx and DTOLx are delayed until the counter has reached the value defined by DTH or DTL. An inverted configuration bit (DTHI and DTLI bit in PWM_CMRx) is provided for each output to invert the dead-time outputs. The following figure shows the waveform of the dead-time generator.

Downloaded from Arrow.com.

Figure 49-9. Complementary Output Waveforms

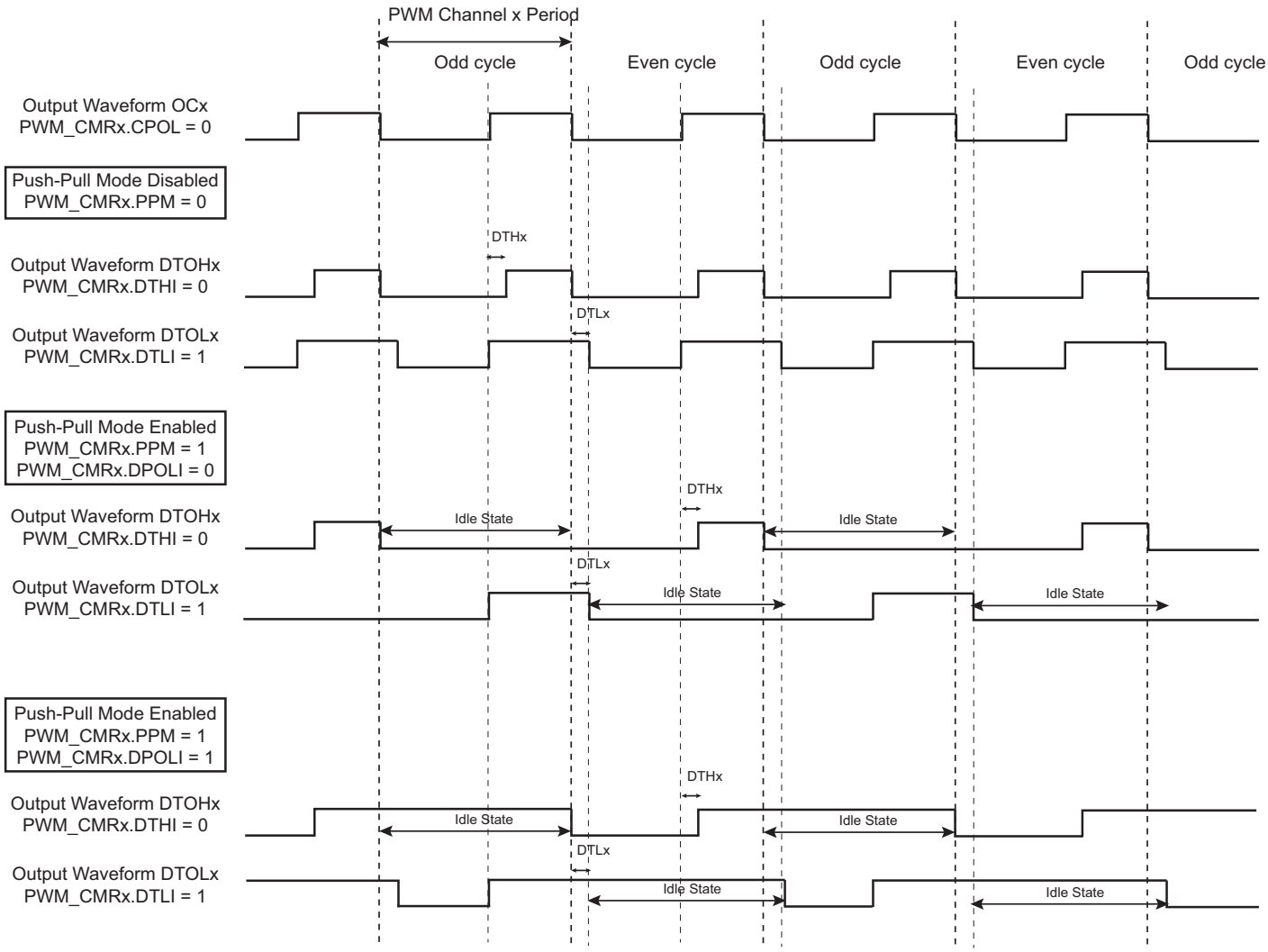


Downloaded from Arrow.com.

PWM Push-Pull Mode

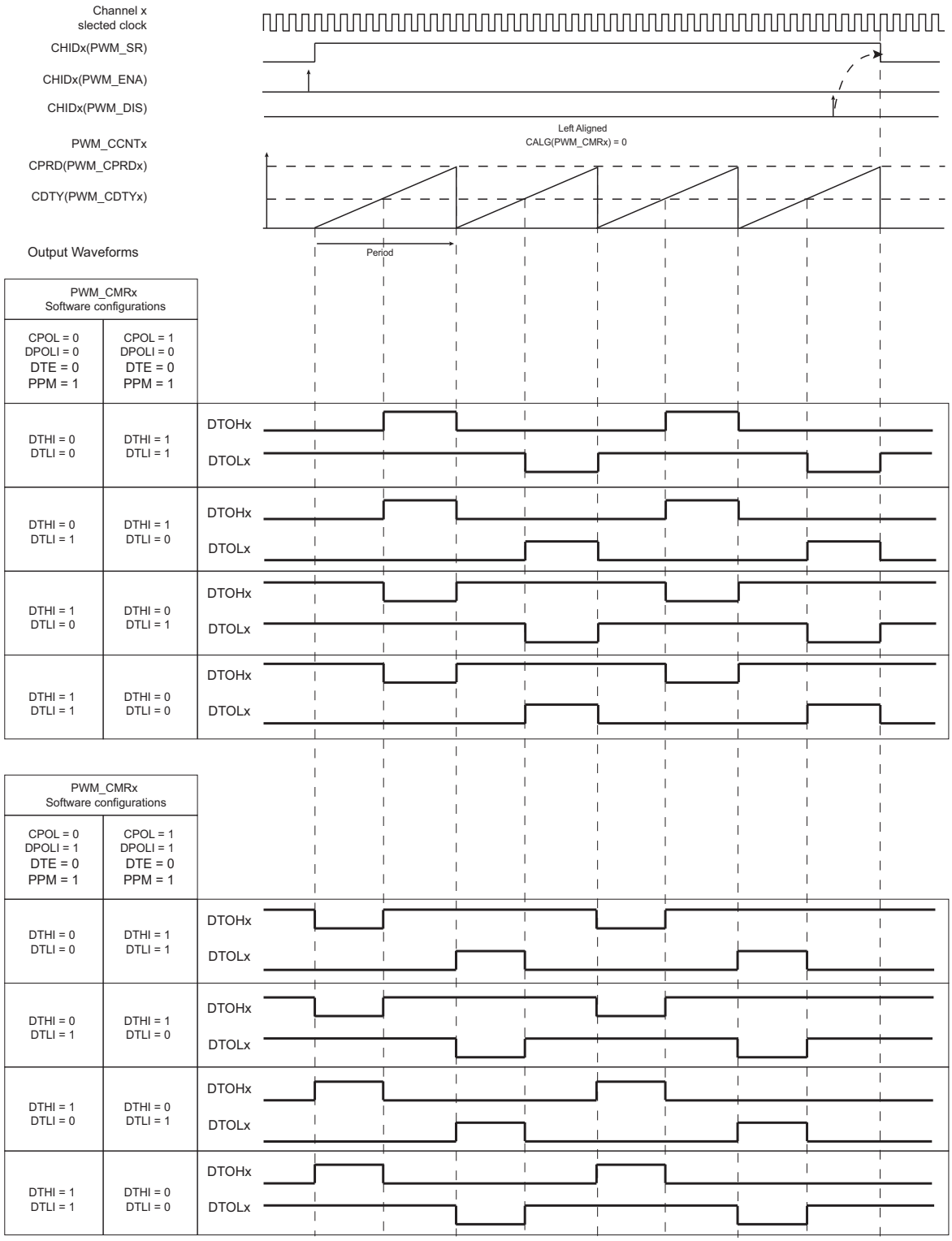
When a PWM channel is configured in Push-Pull mode, the dead-time generator output is managed alternately on each PWM cycle. The polarity of the PWM line during the idle state of the Push-Pull mode is defined by the DPOLI bit in the PWM Channel Mode Register (PWM_CMRx). The Push-Pull mode can be enabled separately on each channel by writing a one to bit PPM in the PWM Channel Mode Register.

Figure 49-10. PWM Push-Pull Mode



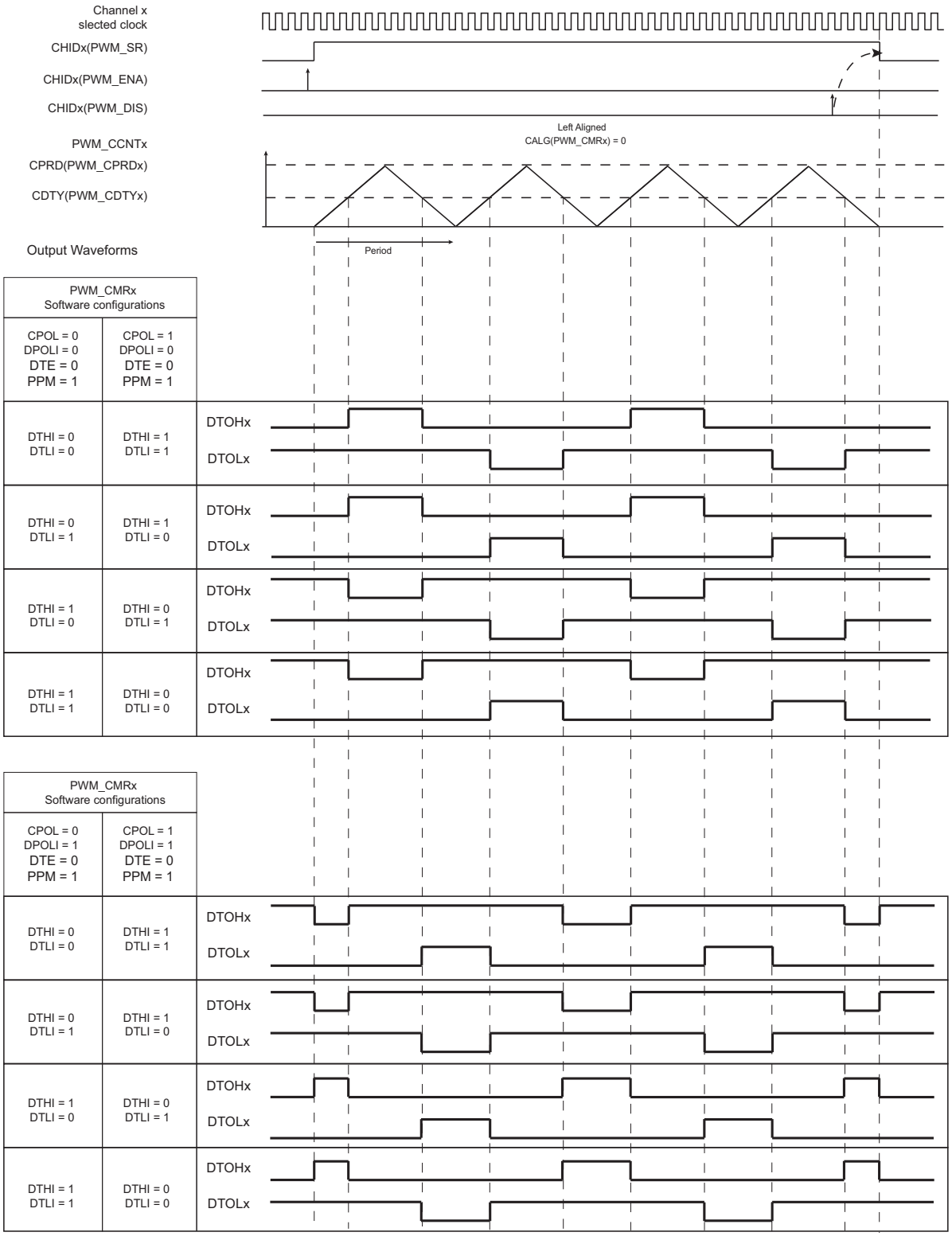
Downloaded from Arrow.com.

Figure 49-11. PWM Push-Pull Waveforms: Left-Aligned Mode



Downloaded from Arrow.com.

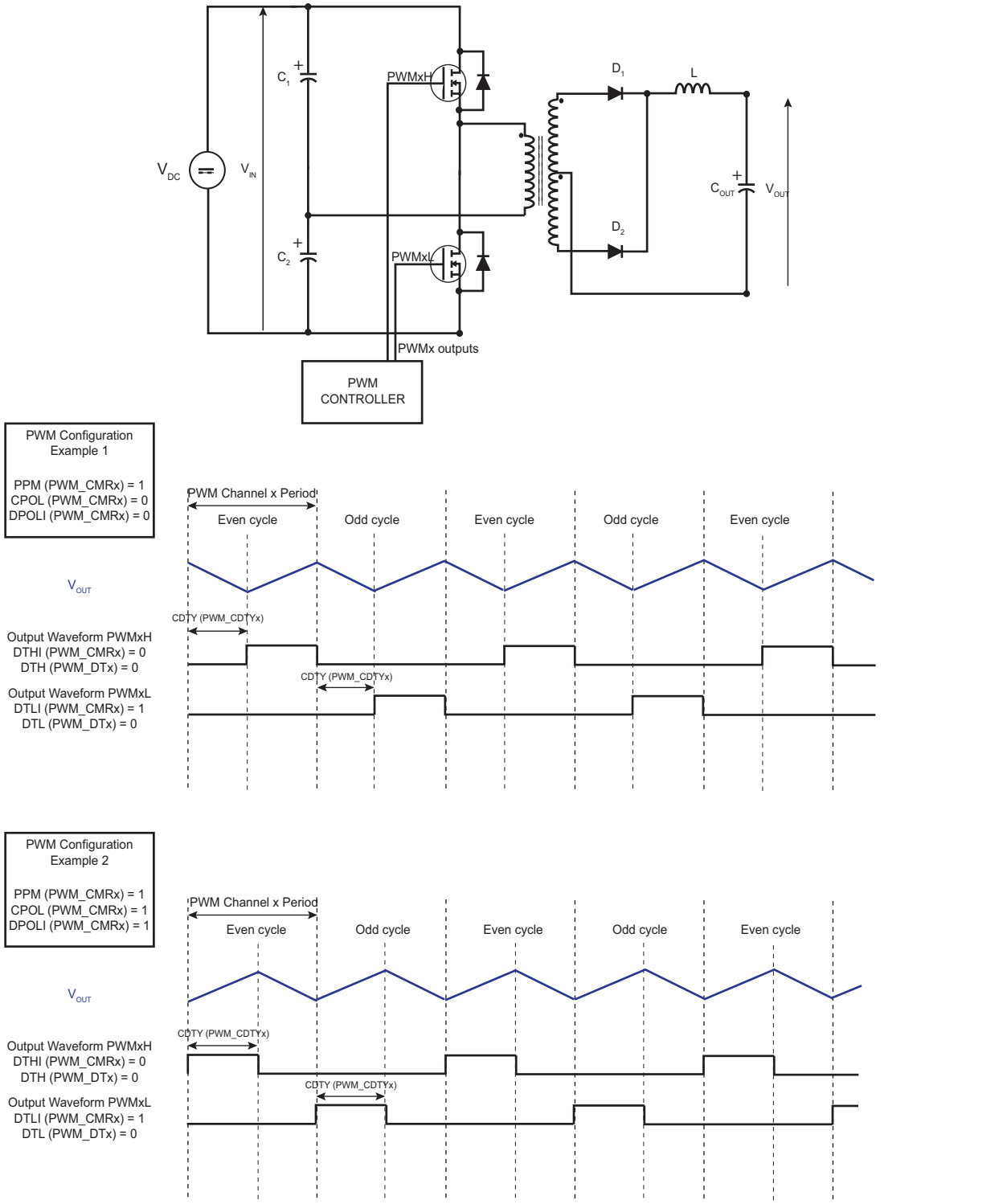
Figure 49-12. PWM Push-Pull Waveforms: Center-Aligned Mode



The PWM Push-Pull mode can be useful in transformer-based power converters, such as a half-bridge converter. The Push-Pull mode prevents the transformer core from being saturated by any direct current.

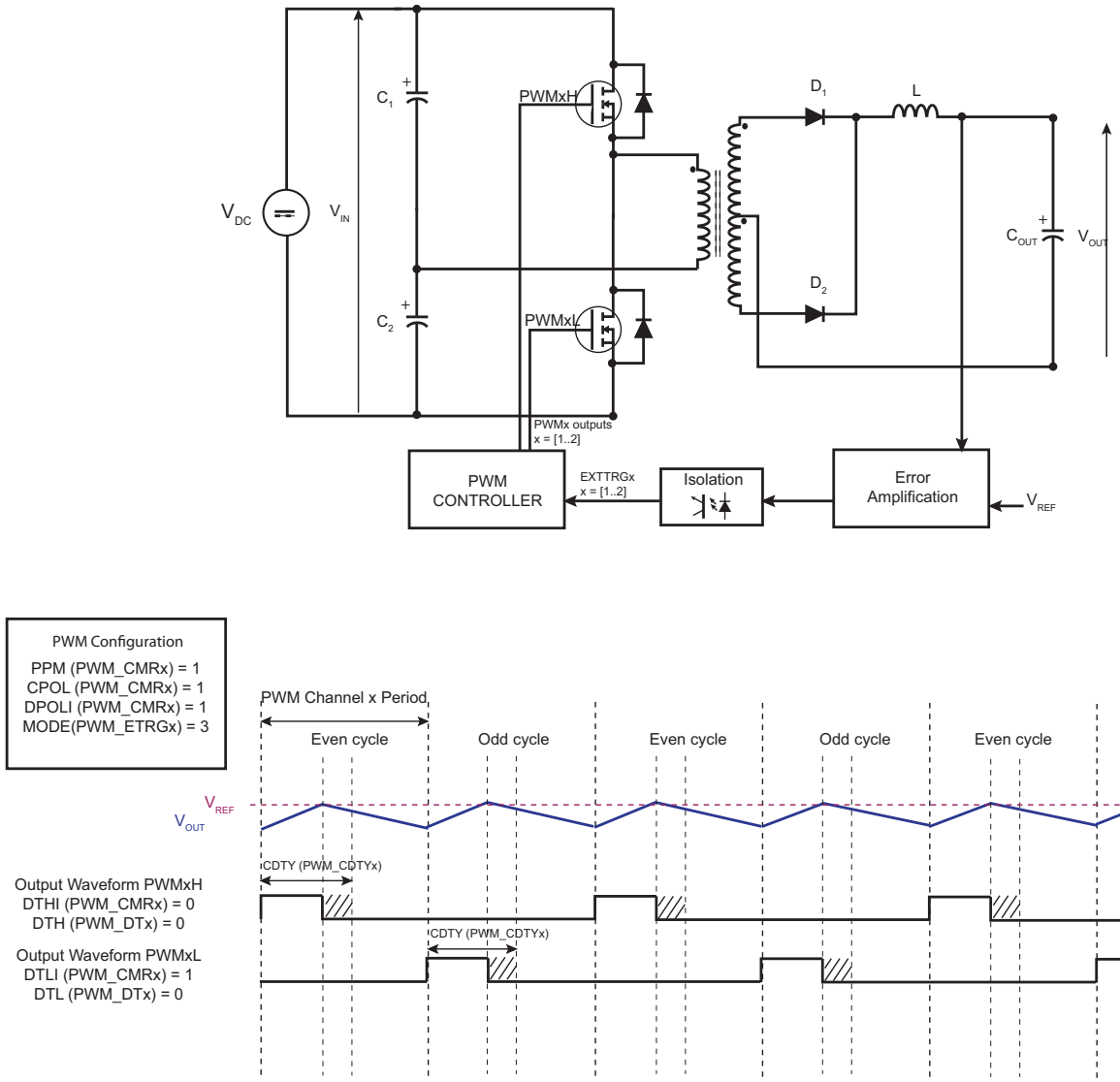
Downloaded from Arrow.com.

Figure 49-13. Half-Bridge Converter Application: No Feedback Regulation



Downloaded from Arrow.com.

Figure 49-14. Half-Bridge Converter Application: Feedback Regulation

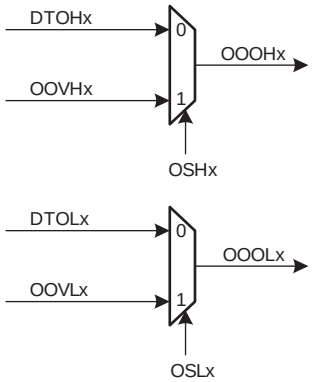


Downloaded from Arrow.com.

49.6.2.6 Output Override

The two complementary outputs DTOHx and DTOLx of the dead-time generator can be forced to a value defined by the software.

Figure 49-15. Override Output Selection



The fields OSHx and OSLx in the PWM Output Selection Register (PWM_OS) allow the outputs of the dead-time generator DTOHx and DTOLx to be overridden by the value defined in the fields OOVHx and OOVLx in the PWM Output Override Value Register (PWM_OOV).

The set registers PWM Output Selection Set Register (PWM_OSS) and PWM Output Selection Set Update Register (PWM_OSSUPD) enable the override of the outputs of a channel regardless of other channels. In the same way, the clear registers PWM Output Selection Clear Register (PWM_OSC) and PWM Output Selection Clear Update Register (PWM_OSCUPD) disable the override of the outputs of a channel regardless of other channels.

By using buffer registers PWM_OSSUPD and PWM_OSCUPD, the output selection of PWM outputs is done synchronously to the channel counter, at the beginning of the next PWM period.

By using registers PWM_OSS and PWM_OSC, the output selection of PWM outputs is done asynchronously to the channel counter, as soon as the register is written.

The value of the current output selection can be read in PWM_OS.

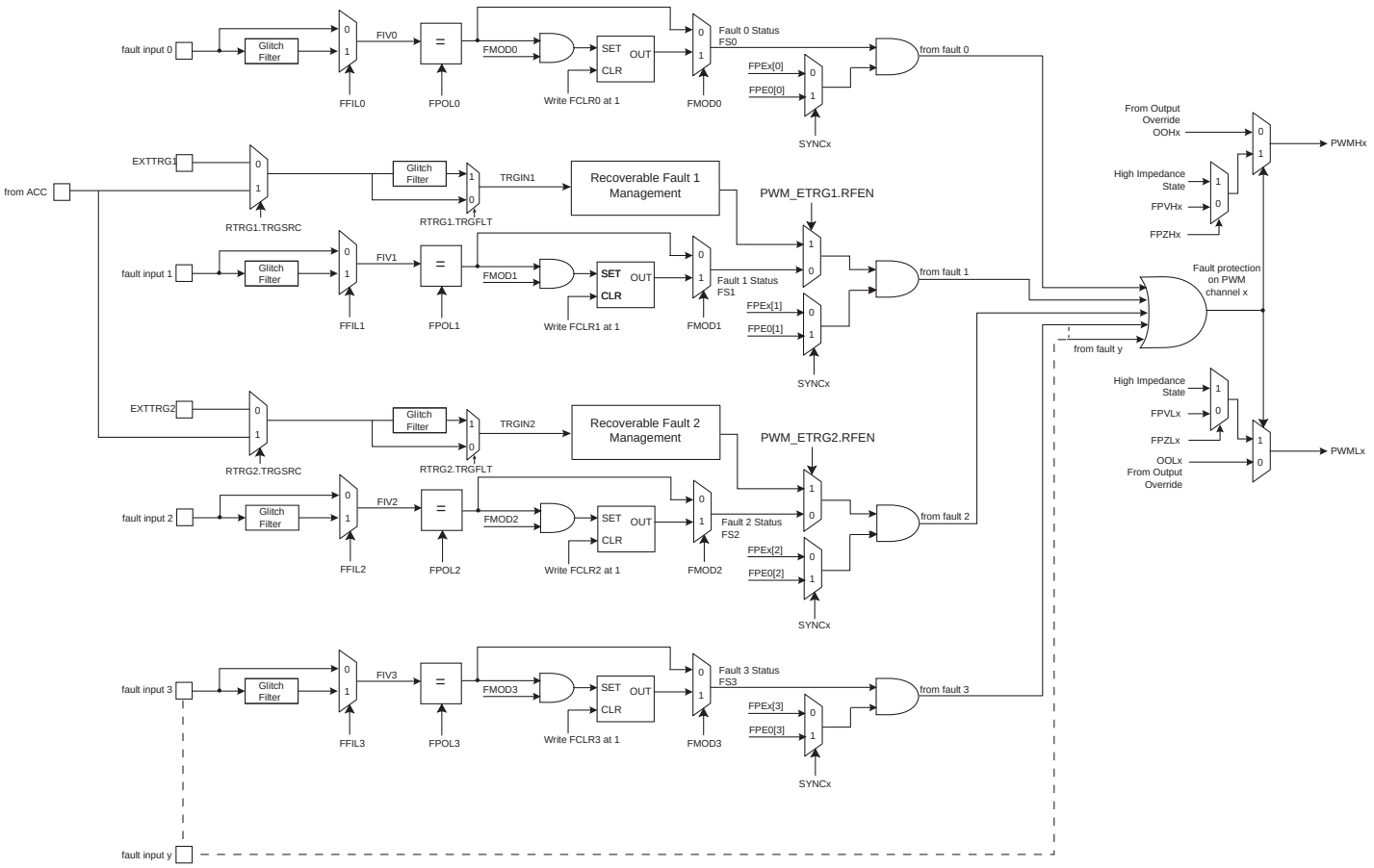
While overriding PWM outputs, the channel counters continue to run, only the PWM outputs are forced to user defined values.

Downloaded from Arrow.com.

49.6.2.7 Fault Protection

8 inputs provide fault protection which can force any of the PWM output pairs to a programmable value. This mechanism has priority over output overriding.

Figure 49-16. Fault Protection



The polarity level of the fault inputs is configured by the FPOL field in the PWM Fault Mode Register (PWM_FMR). For fault inputs coming from internal peripherals such as ADC or Timer Counter, the polarity level must be FPOL = 1. For fault inputs coming from external GPIO pins the polarity level depends on the user's implementation.

The configuration of the Fault Activation mode (FMOD field in PWMC_FMR) depends on the peripheral generating the fault. If the corresponding peripheral does not have “Fault Clear” management, then the FMOD configuration to use must be FMOD = 1, to avoid spurious fault detection. Refer to the corresponding peripheral documentation for details on handling fault generation.

Fault inputs may or may not be glitch-filtered depending on the FFIL field in PWM_FMR. When the filter is activated, glitches on fault inputs with a width inferior to the PWM peripheral clock period are rejected.

A fault becomes active as soon as its corresponding fault input has a transition to the programmed polarity level. If the corresponding bit FMOD is set to ‘0’ in PWM_FMR, the fault remains active as long as the fault input is at this polarity level. If the corresponding FMOD field is set to ‘1’, the fault remains active until the fault input is no longer at this polarity level and until it is cleared by writing the corresponding bit FCLR in the PWM Fault Clear Register (PWM_FCR). In the PWM Fault Status Register (PWM_FSR), the field FIV indicates the current level of the fault inputs and the field FIS indicates whether a fault is currently active.

Each fault can be taken into account or not by the fault protection mechanism in each channel. To be taken into account in the channel x, the fault y must be enabled by the bit FPEx[y] in the PWM Fault Protection Enable

Downloaded from Arrow.com.

registers (PWM_FPE1). However, synchronous channels (see Section 49.6.2.9 “Synchronous Channels”) do not use their own fault enable bits, but those of the channel 0 (bits FPE0[y]).

The fault protection on a channel is triggered when this channel is enabled and when any one of the faults that are enabled for this channel is active. It can be triggered even if the PWM peripheral clock is not running but only by a fault input that is not glitch-filtered.

When the fault protection is triggered on a channel, the fault protection mechanism resets the counter of this channel and forces the channel outputs to the values defined by the fields FPVHx and FPVLx in the PWM Fault Protection Value Register 1 (PWM_FPV) and fields FPZHx/FPZLx in the PWM Fault Protection Value Register 2, as shown in Table 49-5. The output forcing is made asynchronously to the channel counter.

Table 49-5. Forcing Values of PWM Outputs by Fault Protection

FPZH/Lx	FPVH/Lx	Forcing Value of PWMH/Lx
0	0	0
0	1	1
1	–	High impedance state (Hi-Z)

CAUTION:

- To prevent any unexpected activation of the status flag FSy in PWM_FSR, the FMODy bit can be set to ‘1’ only if the FPOLy bit has been previously configured to its final value.
- To prevent any unexpected activation of the Fault Protection on the channel x, the bit FPEx[y] can be set to ‘1’ only if the FPOLy bit has been previously configured to its final value.

If a comparison unit is enabled (see Section 49.6.3 “PWM Comparison Units”) and if a fault is triggered in the channel 0, then the comparison cannot match.

As soon as the fault protection is triggered on a channel, an interrupt (different from the interrupt generated at the end of the PWM period) can be generated but only if it is enabled and not masked. The interrupt is reset by reading the interrupt status register, even if the fault which has caused the trigger of the fault protection is kept active.

Recoverable Fault

The PWM provides a Recoverable Fault mode on fault 1 and 2 (see Figure 49-16).

The recoverable fault signal is an internal signal generated as soon as an external trigger event occurs (see Section 49.6.5 “PWM External Trigger Mode”).

When the fault 1 or 2 is defined as a recoverable fault, the corresponding fault input pin is ignored and bits FFIL1/2, FMOD1/2 and FFIL1/2 are not taken into account.

When PWM_ETRG1.RFEN = 1 and PWM_ETRG1.TRGMODE ≠ 0, the fault 1 is managed as a recoverable fault by the PWMTRG1 input trigger.

When PWM_ETRG2.RFEN = 1 and PWM_ETRG1.TRGMODE ≠ 0, the fault 2 is managed as a recoverable fault by the PWMTRG2 input trigger.

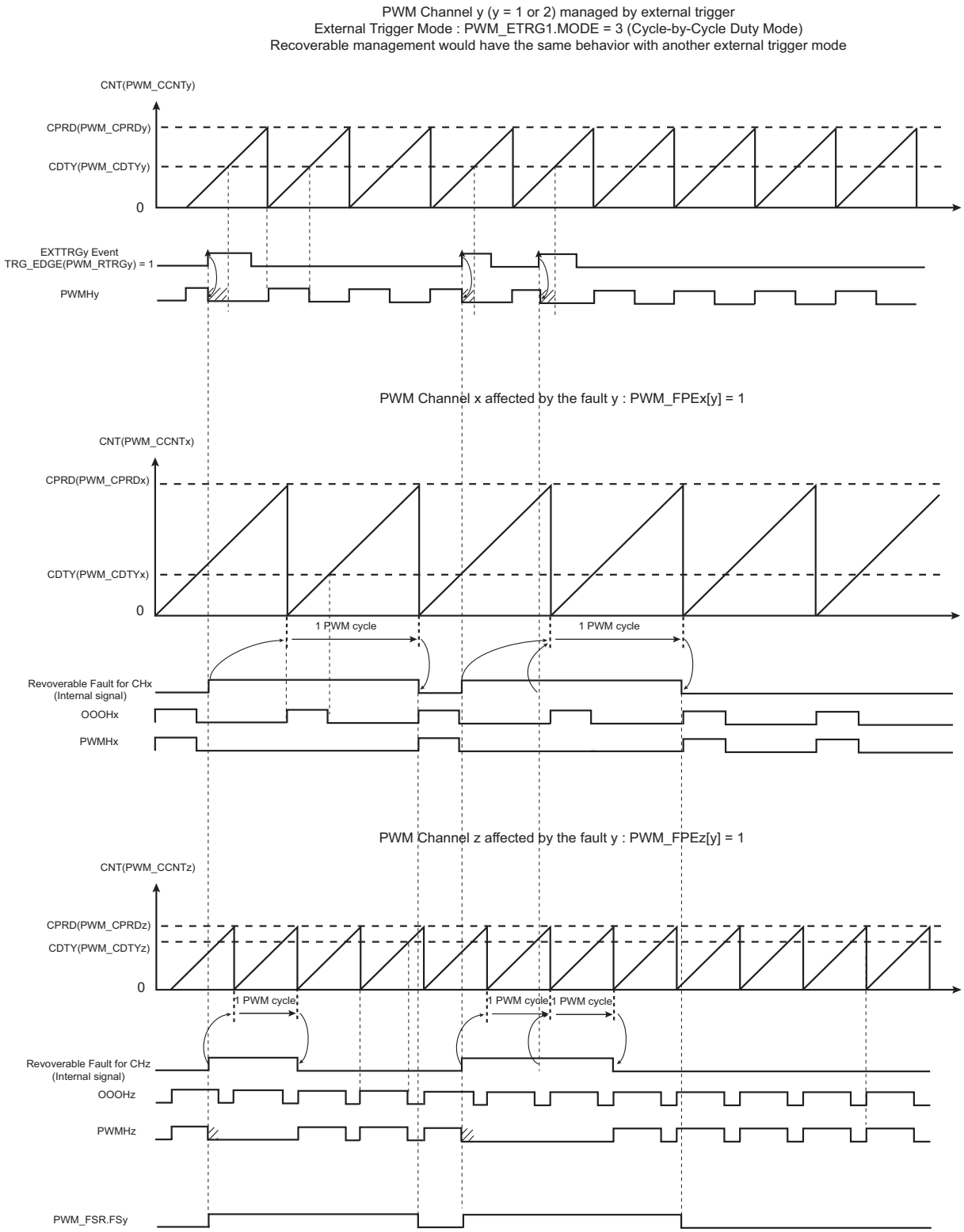
Recoverable fault 1 and 2 can be taken into account by all channels by enabling the bit FPEx[1/2] in the PWM Fault Protection Enable registers (PWM_FPEx). However the synchronous channels (see Section 49.6.2.9 “Synchronous Channels”) do not use their own fault enable bits, but those of the channel 0 (bits FPE0[1/2]).

When a recoverable fault is triggered, the PWM counter of the affected channels is not cleared (unlike in the classic fault protection mechanism) but the channel outputs are forced to the values defined by the fields FPVHx and FPVLx in the PWM Fault Protection Value Register 1 (PWM_FPV), as per Table 49-5. The output forcing is made asynchronously to the channel counter and lasts from the recoverable fault occurrence to the end of the next PWM cycle (if the recoverable fault is no longer present) (see Figure 49-17).

The recoverable fault does not trigger an interrupt. The Fault Status FSy (with y = 1 or 2) is not reported in the PWM Fault Status Register when the fault y is a recoverable fault.

Downloaded from Arrow.com.

Figure 49-17. Recoverable Fault Management



Downloaded from Arrow.com.

49.6.2.8 Spread Spectrum Counter

The PWM macrocell includes a spread spectrum counter allowing the generation of a constantly varying duty cycle on the output PWM waveform (only for the channel 0). This feature may be useful to minimize electromagnetic interference or to reduce the acoustic noise of a PWM driven motor.

This is achieved by varying the effective period in a range defined by a spread spectrum value which is programmed by the field SPRD in the PWM Spread Spectrum Register (PWM_SSPR). The effective period of the output waveform is the value of the spread spectrum counter added to the programmed waveform period CPRD in the PWM Channel Period Register (PWM_CPRD0).

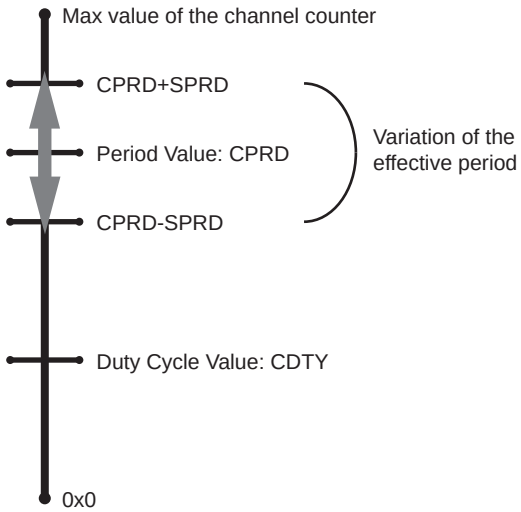
It will cause the effective period to vary from CPRD-SPRD to CPRD+SPRD. This leads to a constantly varying duty cycle on the PWM output waveform because the duty cycle value programmed is unchanged.

The value of the spread spectrum counter can change in two ways depending on the bit SPRDM in PWM_SSPR.

If SPRDM = 0, the Triangular mode is selected. The spread spectrum counter starts to count from -SPRD when the channel 0 is enabled or after reset and counts upwards at each period of the channel counter. When it reaches SPRD, it restarts to count from -SPRD again.

If SPRDM = 1, the Random mode is selected. A new random value is assigned to the spread spectrum counter at each period of the channel counter. This random value is between -SPRD and +SPRD and is uniformly distributed.

Figure 49-18. Spread Spectrum Counter



49.6.2.9 Synchronous Channels

Some channels can be linked together as synchronous channels. They have the same source clock, the same period, the same alignment and are started together. In this way, their counters are synchronized together.

The synchronous channels are defined by the SYNCx bits in the PWM Sync Channels Mode Register (PWM_SCM). Only one group of synchronous channels is allowed.

When a channel is defined as a synchronous channel, the channel 0 is also automatically defined as a synchronous channel. This is because the channel 0 counter configuration is used by all the synchronous channels.

If a channel x is defined as a synchronous channel, the fields/bits for the channel 0 are used instead of those of channel x:

- CPRE in PWM_CMR0 instead of CPRE in PWM_CMRx (same source clock)
- CPRD in PWM_CPRD0 instead of CPRD in PWM_CPRDx (same period)
- CALG in PWM_CMR0 instead of CALG in PWM_CMRx (same alignment)

Downloaded from Arrow.com.

Modifying the fields CPRE, CPRD and CALG of for channels with index greater than 0 has no effect on output waveforms.

Because counters of synchronous channels must start at the same time, they are all enabled together by enabling the channel 0 (by the CHID0 bit in PWM_ENA register). In the same way, they are all disabled together by disabling channel 0 (by the CHID0 bit in PWM_DIS register). However, a synchronous channel x different from channel 0 can be enabled or disabled independently from others (by the CHIDx bit in PWM_ENA and PWM_DIS registers).

Defining a channel as a synchronous channel while it is an asynchronous channel (by writing the bit SYNCx to '1' while it was at '0') is allowed only if the channel is disabled at this time (CHIDx = 0 in PWM_SR). In the same way, defining a channel as an asynchronous channel while it is a synchronous channel (by writing the SYNCx bit to '0' while it was '1') is allowed only if the channel is disabled at this time.

The UPDM field (Update Mode) in the PWM_SCM register selects one of the three methods to update the registers of the synchronous channels:

- Method 1 (UPDM = 0): The period value, the duty-cycle values and the dead-time values must be written by the processor in their respective update registers (respectively PWM_CPRDUPDx, PWM_CDTYUPDx and PWM_DTUPDx).The update is triggered at the next PWM period as soon as the bit UPDULOCK in the PWM Sync Channels Update Control Register (PWM_SCUC) is set to '1' (see "Method 1: Manual write of duty-cycle values and manual trigger of the update").
- Method 2 (UPDM = 1): The period value, the duty-cycle values, the dead-time values and the update period value must be written by the processor in their respective update registers (respectively PWM_CPRDUPDx, PWM_CDTYUPDx and PWM_DTUPD). The update of the period value and of the dead-time values is triggered at the next PWM period as soon as the bit UPDULOCK in the PWM_SCUC register is set to '1'. The update of the duty-cycle values and the update period value is triggered automatically after an update period defined by the field UPR in the PWM Sync Channels Update Period Register (PWM_SCUP) (see "Method 2: Manual write of duty-cycle values and automatic trigger of the update").
- Method 3 (UPDM = 2): Same as Method 2 apart from the fact that the duty-cycle values of ALL synchronous channels are written by the DMA Controller (see "Method 3: Automatic write of duty-cycle values and automatic trigger of the update"). The user can choose to synchronize the DMA Controller transfer request with a comparison match (see Section 49.6.3 "PWM Comparison Units"), by the fields PTRM and PTRCS in the PWM_SCM register. The DMA destination address must be configured to access only the PWM DMA Register (PWM_DMAR). The DMA buffer data structure must consist of sequentially repeated duty cycles. The number of duty cycles in each sequence corresponds to the number of synchronized channels. Duty cycles in each sequence must be ordered from the lowest to the highest channel index. The size of the duty cycle is 16 bits.

Table 49-6. Summary of the Update of Registers of Synchronous Channels

Register	UPDM = 0	UPDM = 1	UPDM = 2
Period Value (PWM_CPRDUPDx)	Write by the processor		
	Update is triggered at the next PWM period as soon as the bit UPDULOCK is set to '1'		
Dead-Time Values (PWM_DTUPDx)	Write by the processor		
	Update is triggered at the next PWM period as soon as the bit UPDULOCK is set to '1'		

Downloaded from Arrow.com.

Table 49-6. Summary of the Update of Registers of Synchronous Channels (Continued)

Register	UPDM = 0	UPDM = 1	UPDM = 2
Duty-Cycle Values (PWM_CDTYUPDx)	Write by the processor	Write by the processor	Write by the DMA Controller
	Update is triggered at the next PWM period as soon as the bit UPDULOCK is set to '1'	Update is triggered at the next PWM period as soon as the update period counter has reached the value UPR	
Update Period Value (PWM_SCUPUPD)	Not applicable	Write by the processor	
	Not applicable	Update is triggered at the next PWM period as soon as the update period counter has reached the value UPR	

Method 1: Manual write of duty-cycle values and manual trigger of the update

In this mode, the update of the period value, the duty-cycle values and the dead-time values must be done by writing in their respective update registers with the processor (respectively PWM_CPRDUPDx, PWM_CDTYUPDx and PWM_DTUPDx).

To trigger the update, the user must use the bit UPDULOCK in the PWM_SCUC register which allows to update synchronously (at the same PWM period) the synchronous channels:

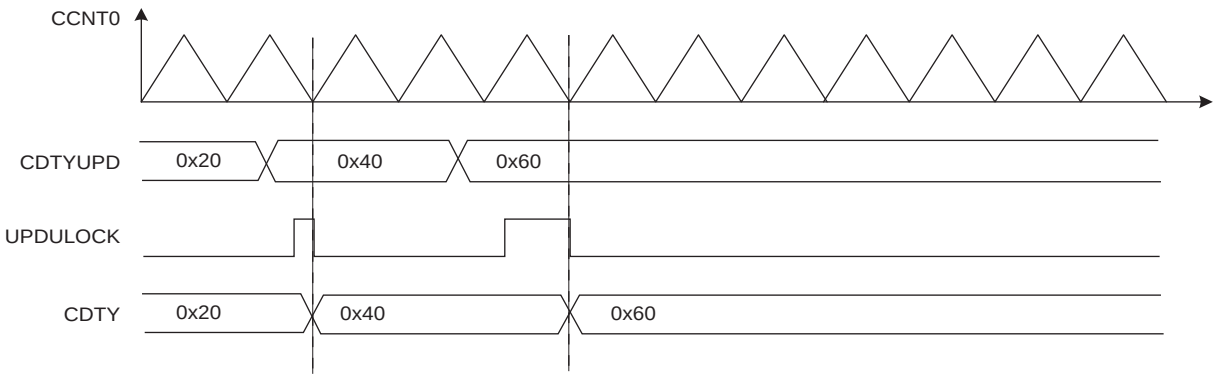
- If the bit UPDULOCK is set to '1', the update is done at the next PWM period of the synchronous channels.
- If the UPDULOCK bit is not set to '1', the update is locked and cannot be performed.

After writing the UPDULOCK bit to '1', it is held at this value until the update occurs, then it is read 0.

Sequence for Method 1:

1. Select the manual write of duty-cycle values and the manual update by setting the UPDM field to '0' in the PWM_SCM register.
2. Define the synchronous channels by the SYNCx bits in the PWM_SCM register.
3. Enable the synchronous channels by writing CHID0 in the PWM_ENA register.
4. If an update of the period value and/or the duty-cycle values and/or the dead-time values is required, write registers that need to be updated (PWM_CPRDUPDx, PWM_CDTYUPDx and PWM_DTUPDx).
5. Set UPDULOCK to '1' in PWM_SCUC.
6. The update of the registers will occur at the beginning of the next PWM period. When the UPDULOCK bit is reset, go to Step 4. for new values.

Figure 49-19. Method 1 (UPDM = 0)



Downloaded from Arrow.com.

Method 2: Manual write of duty-cycle values and automatic trigger of the update

In this mode, the update of the period value, the duty-cycle values, the dead-time values and the update period value must be done by writing in their respective update registers with the processor (respectively PWM_CPRDUPDx, PWM_CDTYUPDx, PWM_DTUPDx and PWM_SCUPUPD).

To trigger the update of the period value and the dead-time values, the user must use the bit UPDULOCK in the PWM_SCUC register, which updates synchronously (at the same PWM period) the synchronous channels:

- If the bit UPDULOCK is set to '1', the update is done at the next PWM period of the synchronous channels.
- If the UPDULOCK bit is not set to '1', the update is locked and cannot be performed.

After writing the UPDULOCK bit to '1', it is held at this value until the update occurs, then it is read 0.

The update of the duty-cycle values and the update period is triggered automatically after an update period.

To configure the automatic update, the user must define a value for the update period by the UPR field in the PWM_SCUP register. The PWM controller waits UPR+1 period of synchronous channels before updating automatically the duty values and the update period value.

The status of the duty-cycle value write is reported in the PWM Interrupt Status Register 2 (PWM_ISR2) by the following flags:

- WRDY: this flag is set to '1' when the PWM Controller is ready to receive new duty-cycle values and a new update period value. It is reset to '0' when the PWM_ISR2 register is read.

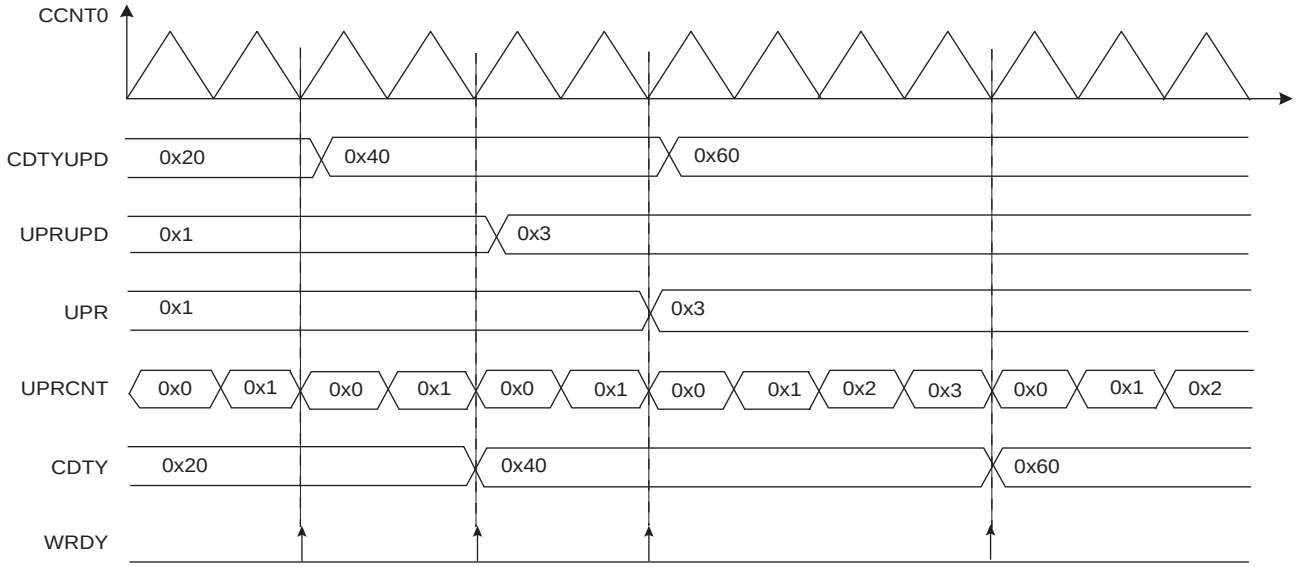
Depending on the interrupt mask in the PWM Interrupt Mask Register 2 (PWM_IMR2), an interrupt can be generated by these flags.

Sequence for Method 2:

1. Select the manual write of duty-cycle values and the automatic update by setting the field UPDM to '1' in the PWM_SCM register
2. Define the synchronous channels by the bits SYNCx in the PWM_SCM register.
3. Define the update period by the field UPR in the PWM_SCUP register.
4. Enable the synchronous channels by writing CHID0 in the PWM_ENA register.
5. If an update of the period value and/or of the dead-time values is required, write registers that need to be updated (PWM_CPRDUPDx, PWM_DTUPDx), else go to Step 8.
6. Set UPDULOCK to '1' in PWM_SCUC.
7. The update of these registers will occur at the beginning of the next PWM period. At this moment the bit UPDULOCK is reset, go to Step 5. for new values.
8. If an update of the duty-cycle values and/or the update period is required, check first that write of new update values is possible by polling the flag WRDY (or by waiting for the corresponding interrupt) in PWM_ISR2.
9. Write registers that need to be updated (PWM_CDTYUPDx, PWM_SCUPUPD).
10. The update of these registers will occur at the next PWM period of the synchronous channels when the Update Period is elapsed. Go to Step 8. for new values.

Downloaded from Arrow.com.

Figure 49-20. Method 2 (UPDM = 1)



Method 3: Automatic write of duty-cycle values and automatic trigger of the update

In this mode, the update of the duty cycle values is made automatically by the DMA Controller. The update of the period value, the dead-time values and the update period value must be done by writing in their respective update registers with the processor (respectively PWM_CPRDUPDx, PWM_DTUPDx and PWM_SCUPUPD).

To trigger the update of the period value and the dead-time values, the user must use the bit UPDULOCK which allows to update synchronously (at the same PWM period) the synchronous channels:

- If the bit UPDULOCK is set to ‘1’, the update is done at the next PWM period of the synchronous channels.
- If the UPDULOCK bit is not set to ‘1’, the update is locked and cannot be performed.

After writing the UPDULOCK bit to ‘1’, it is held at this value until the update occurs, then it is read 0.

The update of the duty-cycle values and the update period value is triggered automatically after an update period.

To configure the automatic update, the user must define a value for the Update Period by the field UPR in the PWM_SCUP register. The PWM controller waits UPR+1 periods of synchronous channels before updating automatically the duty values and the update period value.

Using the DMA Controller removes processor overhead by reducing its intervention during the transfer. This significantly reduces the number of clock cycles required for a data transfer, which improves microcontroller performance.

The DMA Controller must write the duty-cycle values in the synchronous channels index order. For example if the channels 0, 1 and 3 are synchronous channels, the DMA Controller must write the duty-cycle of the channel 0 first, then the duty-cycle of the channel 1, and finally the duty-cycle of the channel 3.

The status of the DMA Controller transfer is reported in PWM_ISR2 by the following flags:

- WRDY: this flag is set to ‘1’ when the PWM Controller is ready to receive new duty-cycle values and a new update period value. It is reset to ‘0’ when PWM_ISR2 is read. The user can choose to synchronize the WRDY flag and the DMA Controller transfer request with a comparison match (see Section 49.6.3 “PWM Comparison Units”), by the fields PTRM and PTRCS in the PWM_SCM register.
- UNRE: this flag is set to ‘1’ when the update period defined by the UPR field has elapsed while the whole data has not been written by the DMA Controller. It is reset to ‘0’ when PWM_ISR2 is read.

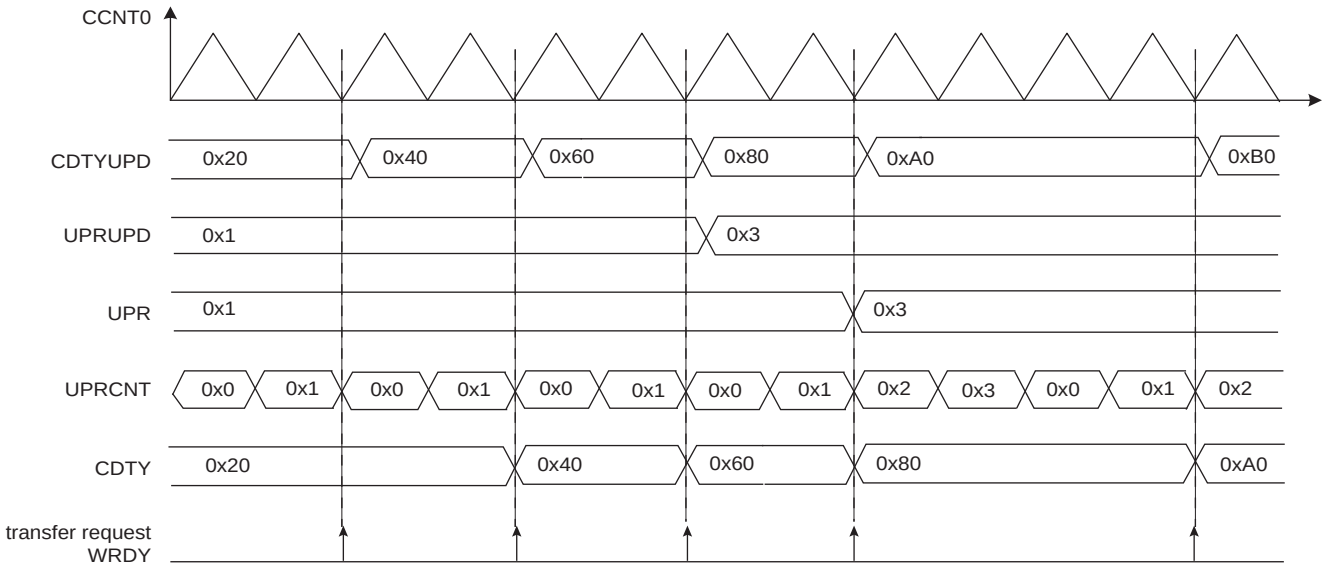
Depending on the interrupt mask in PWM_IMR2, an interrupt can be generated by these flags.

Sequence for Method 3:

Downloaded from Arrow.com.

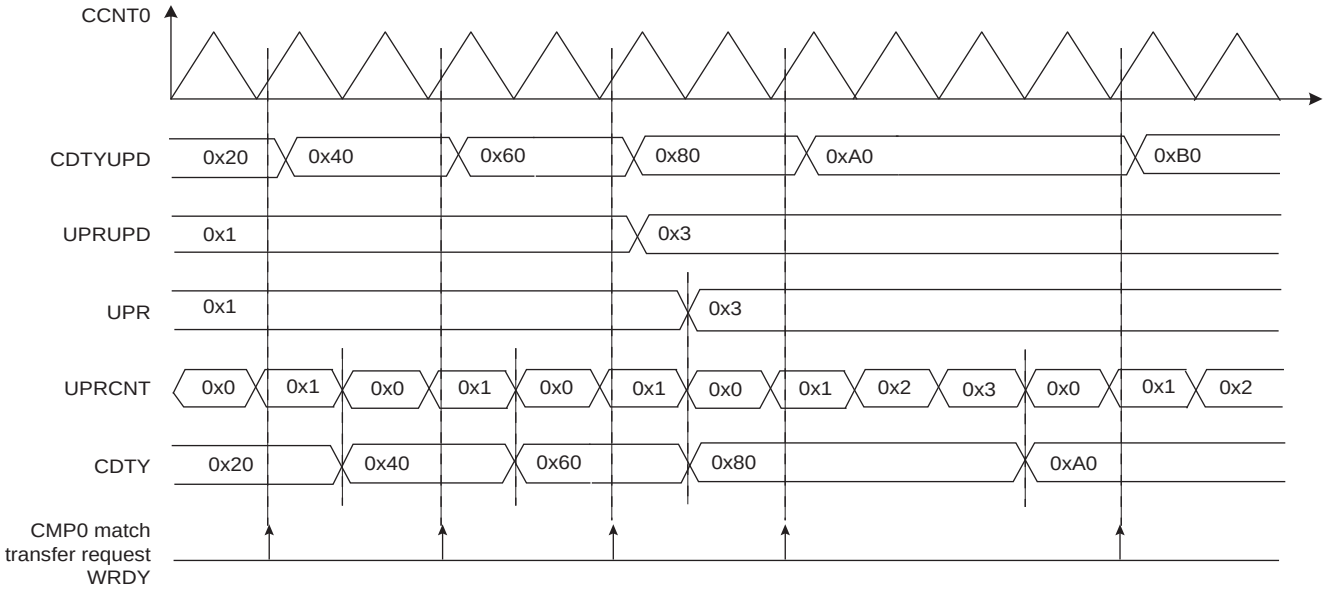
1. Select the automatic write of duty-cycle values and automatic update by setting the field UPDM to 2 in the PWM_SCM register.
2. Define the synchronous channels by the bits SYNCx in the PWM_SCM register.
3. Define the update period by the field UPR in the PWM_SCUP register.
4. Define when the WRDY flag and the corresponding DMA Controller transfer request must be set in the update period by the PTRM bit and the PTRCS field in the PWM_SCM register (at the end of the update period or when a comparison matches).
5. Define the DMA Controller transfer settings for the duty-cycle values and enable it in the DMA Controller registers
6. Enable the synchronous channels by writing CHID0 in the PWM_ENA register.
7. If an update of the period value and/or of the dead-time values is required, write registers that need to be updated (PWM_CPRDUPDx, PWM_DTUPDx), else go to Step 10.
8. Set UPDULOCK to '1' in PWM_SCUC.
9. The update of these registers will occur at the beginning of the next PWM period. At this moment the bit UPDULOCK is reset, go to Step 7. for new values.
10. If an update of the update period value is required, check first that write of a new update value is possible by polling the flag WRDY (or by waiting for the corresponding interrupt) in PWM_ISR2, else go to Step 13.
11. Write the register that needs to be updated (PWM_SCUPUPD).
12. The update of this register will occur at the next PWM period of the synchronous channels when the Update Period is elapsed. Go to Step 10. for new values.
13. Wait for the DMA status flag indicating that the buffer transfer is complete. If the transfer has ended, define a new DMA transfer for new duty-cycle values. Go to Step 5.

Figure 49-21. Method 3 (UPDM = 2 and PTRM = 0)



Downloaded from Arrow.com.

Figure 49-22. Method 3 (UPDM = 2 and PTRM = 1 and PTRCS = 0)



49.6.2.10 Update Time for Double-Buffering Registers

All channels integrate a double-buffering system in order to prevent an unexpected output waveform while modifying the period, the spread spectrum value, the polarity, the duty-cycle, the dead-times, the output override, and the synchronous channels update period.

This double-buffering system comprises the following update registers:

- PWM Sync Channels Update Period Update Register
- PWM Output Selection Set Update Register
- PWM Output Selection Clear Update Register
- PWM Spread Spectrum Update Register
- PWM Channel Duty Cycle Update Register
- PWM Channel Period Update Register
- PWM Channel Dead Time Update Register
- PWM Channel Mode Update Register

When one of these update registers is written to, the write is stored, but the values are updated only at the next PWM period border. In Left-aligned mode (CALG = 0), the update occurs when the channel counter reaches the period value CPRD. In Center-aligned mode, the update occurs when the channel counter value is decremented and reaches the 0 value.

In Center-aligned mode, it is possible to trigger the update of the polarity and the duty-cycle at the next half period border. This mode concerns the following update registers:

- PWM Channel Duty Cycle Update Register
- PWM Channel Mode Update Register

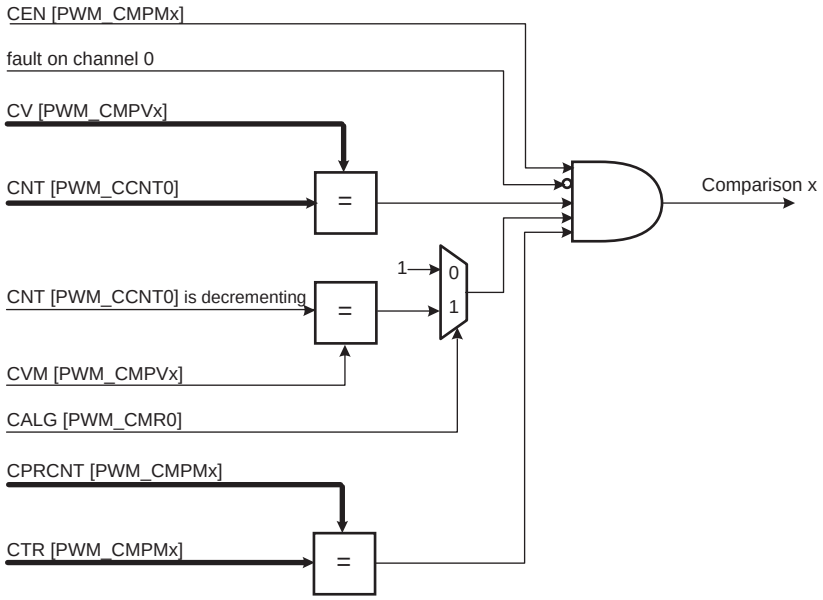
The update occurs at the first half period following the write of the update register (either when the channel counter value is incrementing and reaches the period value CPRD, or when the channel counter value is decrementing and reaches the 0 value). To activate this mode, the user must write a one to the bit UPDS in the PWM Channel Mode Register.

Downloaded from Arrow.com.

49.6.3 PWM Comparison Units

The PWM provides 8 independent comparison units able to compare a programmed value with the current value of the channel 0 counter (which is the channel counter of all synchronous channels, Section 49.6.2.9 “Synchronous Channels”). These comparisons are intended to generate pulses on the event lines (used to synchronize ADC, see Section 49.6.4 “PWM Event Lines”), to generate software interrupts and to trigger DMA Controller transfer requests for the synchronous channels (see “Method 3: Automatic write of duty-cycle values and automatic trigger of the update”).

Figure 49-23. Comparison Unit Block Diagram



The comparison x matches when it is enabled by the bit CEN in the PWM Comparison x Mode Register (PWM_CMPMx for the comparison x) and when the counter of the channel 0 reaches the comparison value defined by the field CV in PWM Comparison x Value Register (PWM_CMPVx for the comparison x). If the counter of the channel 0 is center-aligned (CALG = 1 in PWM Channel Mode Register), the bit CVM in PWM_CMPVx defines if the comparison is made when the counter is counting up or counting down (in Left-alignment mode CALG = 0, this bit is useless).

If a fault is active on the channel 0, the comparison is disabled and cannot match (see Section 49.6.2.7 “Fault Protection”).

The user can define the periodicity of the comparison x by the fields CTR and CPR in PWM_CMPMx. The comparison is performed periodically once every CPR+1 periods of the counter of the channel 0, when the value of the comparison period counter CPRCNT in PWM_CMPMx reaches the value defined by CTR. CPR is the maximum value of the comparison period counter CPRCNT. If CPR = CTR = 0, the comparison is performed at each period of the counter of the channel 0.

The comparison x configuration can be modified while the channel 0 is enabled by using the PWM Comparison x Mode Update Register (PWM_CMPMUPDx registers for the comparison x). In the same way, the comparison x value can be modified while the channel 0 is enabled by using the PWM Comparison x Value Update Register (PWM_CMPVUPDx registers for the comparison x).

The update of the comparison x configuration and the comparison x value is triggered periodically after the comparison x update period. It is defined by the field CUPR in PWM_CMPMx. The comparison unit has an update period counter independent from the period counter to trigger this update. When the value of the comparison update period counter CUPRCNT (in PWM_CMPMx) reaches the value defined by CUPR, the update is triggered.

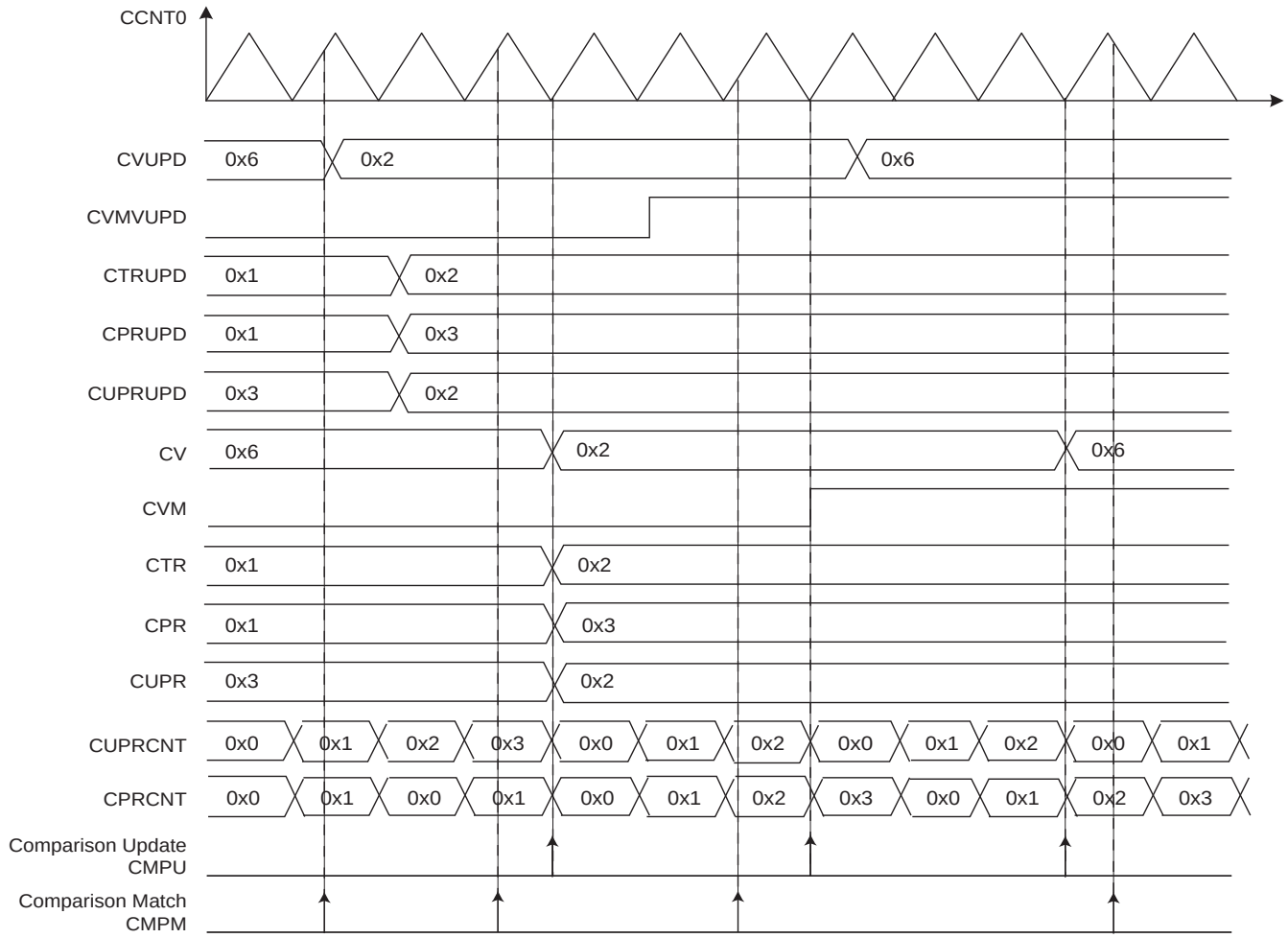
Downloaded from Arrow.com.

The comparison x update period CUPR itself can be updated while the channel 0 is enabled by using the PWM_CMPMUPDx register.

CAUTION: The write of PWM_CMPVUPDx must be followed by a write of PWM_CMPMUPDx.

The comparison match and the comparison update can be source of an interrupt, but only if it is enabled and not masked. These interrupts can be enabled by the PWM Interrupt Enable Register 2 and disabled by the PWM Interrupt Disable Register 2. The comparison match interrupt and the comparison update interrupt are reset by reading the PWM Interrupt Status Register 2.

Figure 49-24. Comparison Waveform



49.6.4 PWM Event Lines

The PWM provides 2 independent event lines intended to trigger actions in other peripherals (e.g., for the Analog-to-Digital Converter (ADC)).

A pulse (one cycle of the peripheral clock) is generated on an event line, when at least one of the selected comparisons is matching. The comparisons can be selected or unselected independently by the CSEL bits in the PWM Event Line x Register (PWM_ELMRx for the Event Line x).

An example of event generation is provided in Figure 49-26.

Downloaded from Arrow.com.

Figure 49-25. Event Line Block Diagram

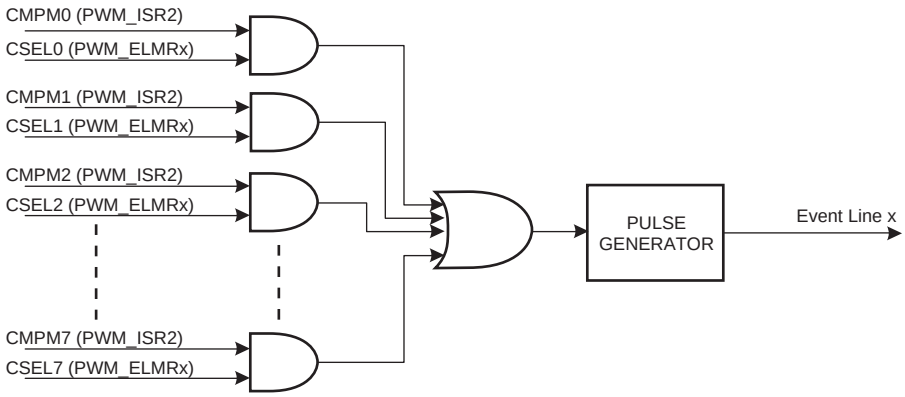
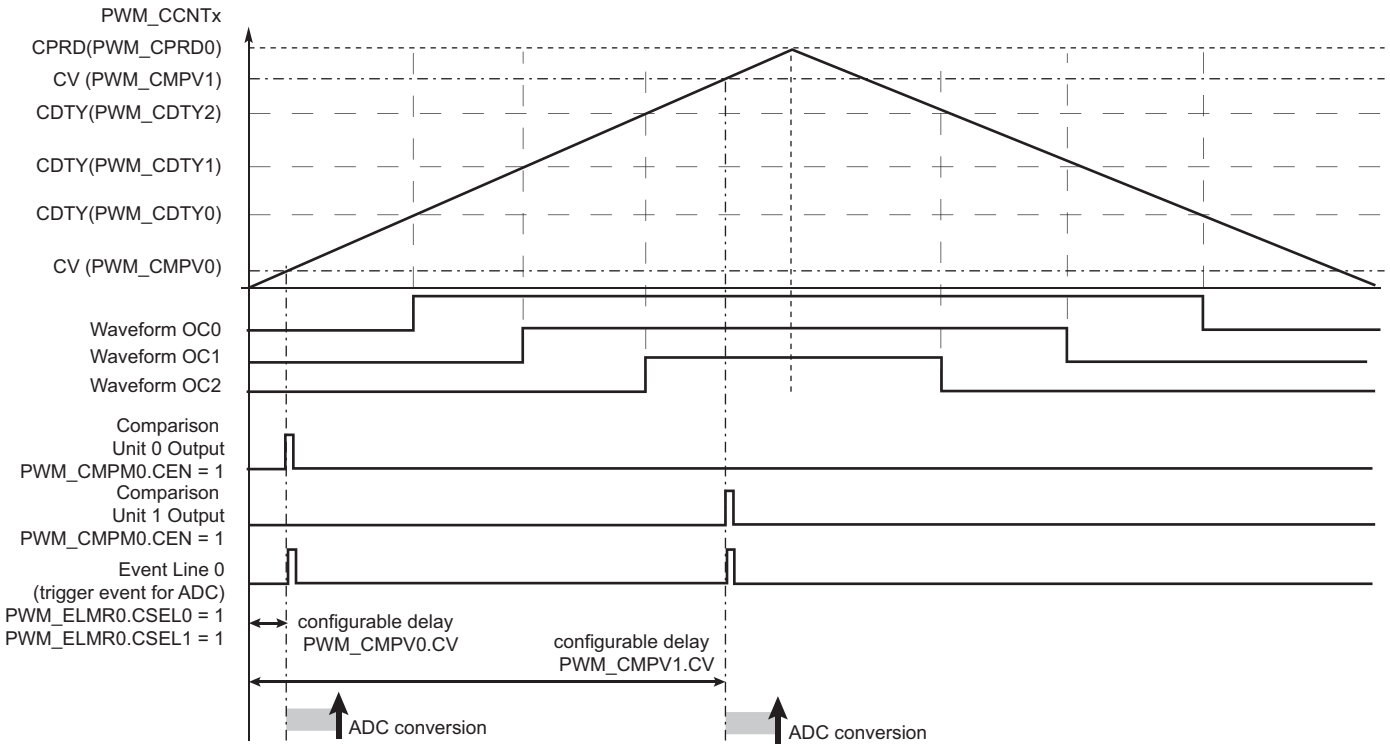


Figure 49-26. Event Line Generation Waveform (Example)



Downloaded from Arrow.com.

49.6.5 PWM External Trigger Mode

The PWM channels 1 and 2 can be configured to use an external trigger for generating specific PWM signals. The external trigger source can be selected through the bit TRGSRC of the PWM External Trigger Register (see Table 49-7).

Table 49-7. External Event Source Selection

Channel	Trigger Source Selection	Trigger Source
1	PWM_ETRG1.TRGSRC = 0	From PWMTRG1 input
	PWM_ETRG1.TRGSRC = 1	From Analog Comparator Controller
2	PWM_ETRG2.TRGSRC = 0	From PWMTRG2 input
	PWM_ETRG2.TRGSRC = 1	From Analog Comparator Controller

Each external trigger source can be filtered by writing a one to the TRGFILT bit in the corresponding PWM External Trigger Register (PWM_ETRGx).

Each time an external trigger event is detected, the corresponding PWM channel counter value is stored in the MAXCNT field of the PWM_ETRGx register if it is greater than the previously stored value. Reading the PWM_ETRGx register will clear the MAXCNT value.

Three different modes are available for channels 1 and 2 depending on the value of the TRGMODE field of the PWM_ETRGx register:

- TRGMODE = 1: External PWM Reset Mode (see Section 49.6.5.1 “External PWM Reset Mode”)
- TRGMODE = 2: External PWM Start Mode (see Section 49.6.5.2 “External PWM Start Mode”)
- TRGMODE = 3: Cycle-By-Cycle Duty Mode (see Section 49.6.5.3 “Cycle-By-Cycle Duty Mode”)

This feature is disabled when TRGMODE = 0.

This feature should only be enabled if the corresponding channel is left-aligned (CALG = 0 in PWM Channel Mode Register of channel 1 or 2) and not managed as a synchronous channel (SYNCx = 0 in PWM Sync Channels Mode Register where x = 1 or 2). Programming the channel to be center-aligned or synchronous while TRGMODE is not 0 could lead to unexpected behavior.

Downloaded from Arrow.com.

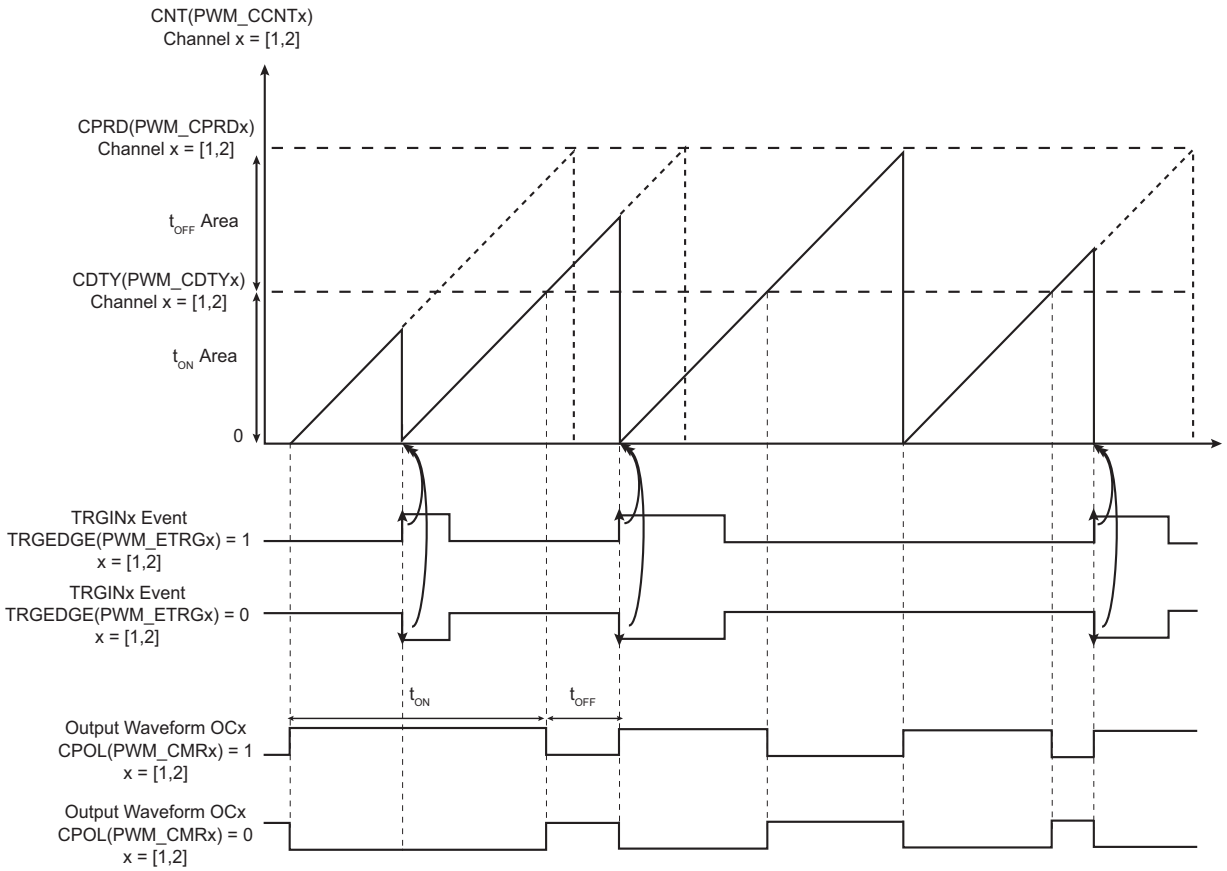
49.6.5.1 External PWM Reset Mode

External PWM Reset mode is selected by programming TRGMODE = 1 in the PWM_ETRGx register.

In this mode, when an edge is detected on the PWMTRGx input, the internal PWM counter is cleared and a new PWM cycle is restarted. The edge polarity can be selected by programming the TRGEDGE bit in the PWM_ETRGx register. If no trigger event is detected when the internal channel counter has reached the CPRD value in the PWM Channel Period Register, the internal counter is cleared and a new PWM cycle starts.

Note that this mode does not guarantee a constant t_{ON} or t_{OFF} time.

Figure 49-27. External PWM Reset Mode



Downloaded from Arrow.com.

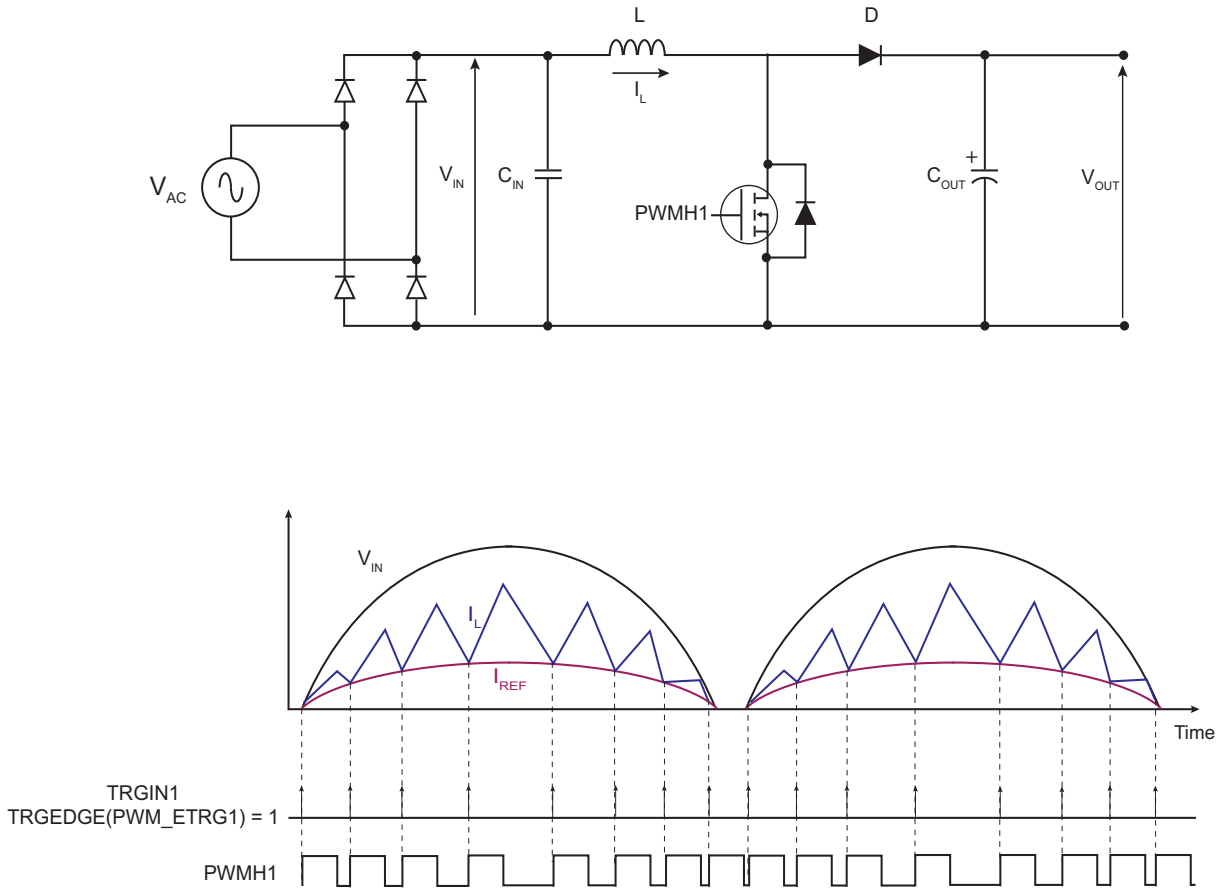
Application Example

The external PWM Reset mode can be used in power factor correction applications.

In the example below, the external trigger input is the PWMTRG1 (therefore the PWM channel used for regulation is the channel 1). The PWM channel 1 period (CPRD in the PWM Channel Period Register of the channel 1) must be programmed so that the TRGIN1 event always triggers before the PWM channel 1 period elapses.

In Figure 49-28, an external circuit (not shown) is required to sense the inductor current I_L. The internal PWM counter of the channel 1 is cleared when the inductor current falls below a specific threshold (I_{REF}). This starts a new PWM period and increases the inductor current.

Figure 49-28. External PWM Reset Mode: Power Factor Correction Application



Downloaded from Arrow.com.

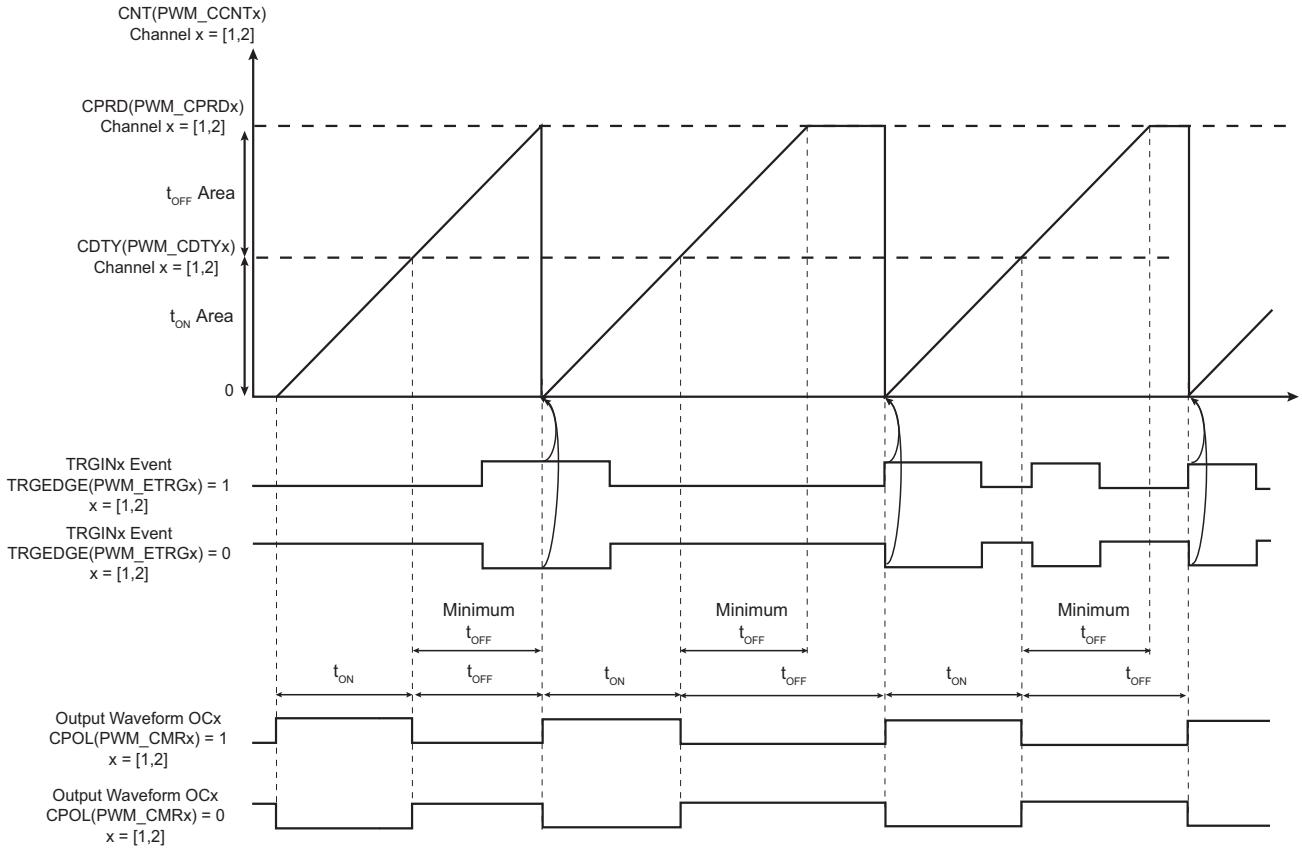
49.6.5.2 External PWM Start Mode

External PWM Start mode is selected by programming TRGMODE = 2 in the PWM_ETRGx register.

In this mode, the internal PWM counter can only be reset once it has reached the CPRD value in the PWM Channel Period Register and when the correct level is detected on the corresponding external trigger input. Both conditions have to be met to start a new PWM period. The active detection level is defined by the bit TRGEDGE of the PWM_ETRGx register.

Note that this mode guarantees a constant t_{ON} time and a minimum t_{OFF} time.

Figure 49-29. External PWM Start Mode



Downloaded from Arrow.com.

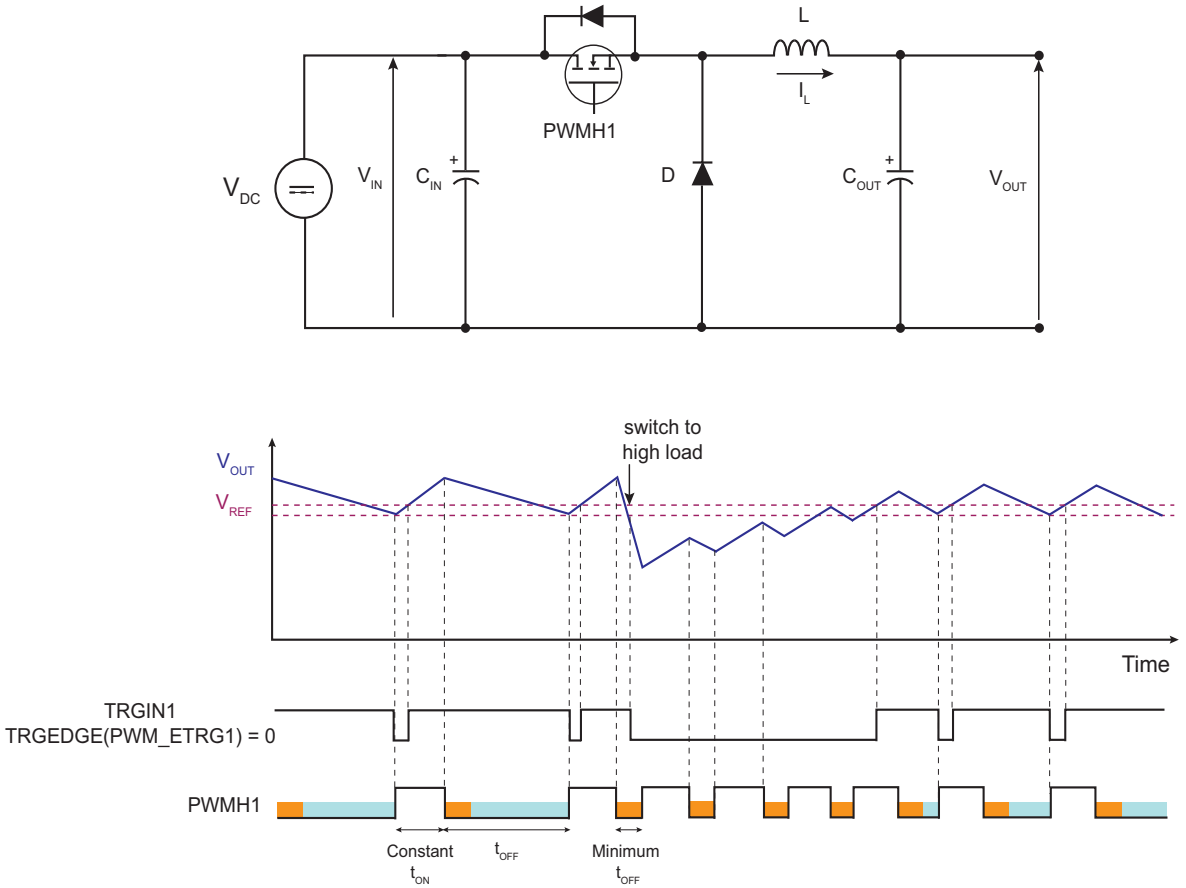
Application Example

The external PWM Start mode generates a modulated frequency PWM signal with a constant active level duration (t_{ON}) and a minimum inactive level duration (minimum t_{OFF}).

The t_{ON} time is defined by the CDTY value in the PWM Channel Duty Cycle Register. The minimum t_{OFF} time is defined by CDTY - CPRD (PWM Channel Period Register). This mode can be useful in Buck DC/DC Converter applications.

When the output voltage V_{OUT} is above a specific threshold (Vref), the PWM inactive level is maintained as long as V_{OUT} remains above this threshold. If V_{OUT} is below this specific threshold, this mode guarantees a minimum t_{OFF} time required for MOSFET driving (see Figure 49-30).

Figure 49-30. External PWM Start Mode: Buck DC/DC Converter



Downloaded from Arrow.com.

49.6.5.3 Cycle-By-Cycle Duty Mode

Description

Cycle-by-cycle duty mode is selected by programming TRGMODE = 3 in PWM_ETRGx.

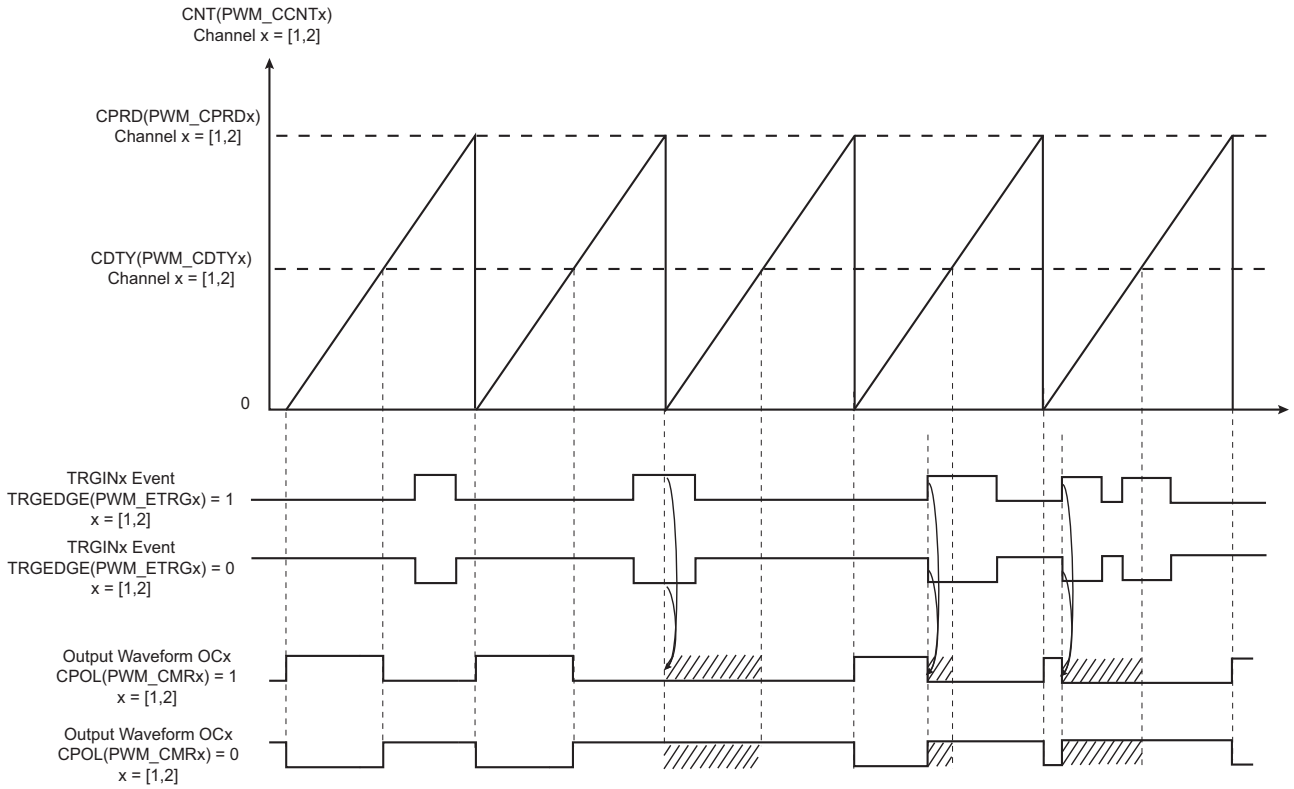
In this mode, the PWM frequency is constant and is defined by the CPRD value in the PWM Channel Period Register.

An external trigger event has no effect on the PWM output if it occurs while the internal PWM counter value is above the CDTY value of the PWM Channel Duty Cycle Register.

If the internal PWM counter value is below the value of CDTY of the PWM Channel Duty Cycle Register, an external trigger event makes the PWM output inactive.

The external trigger event can be detected on rising or falling edge according to the TRGEDGE bit in PWM_ETRGx.

Figure 49-31. Cycle-By-Cycle Duty Mode



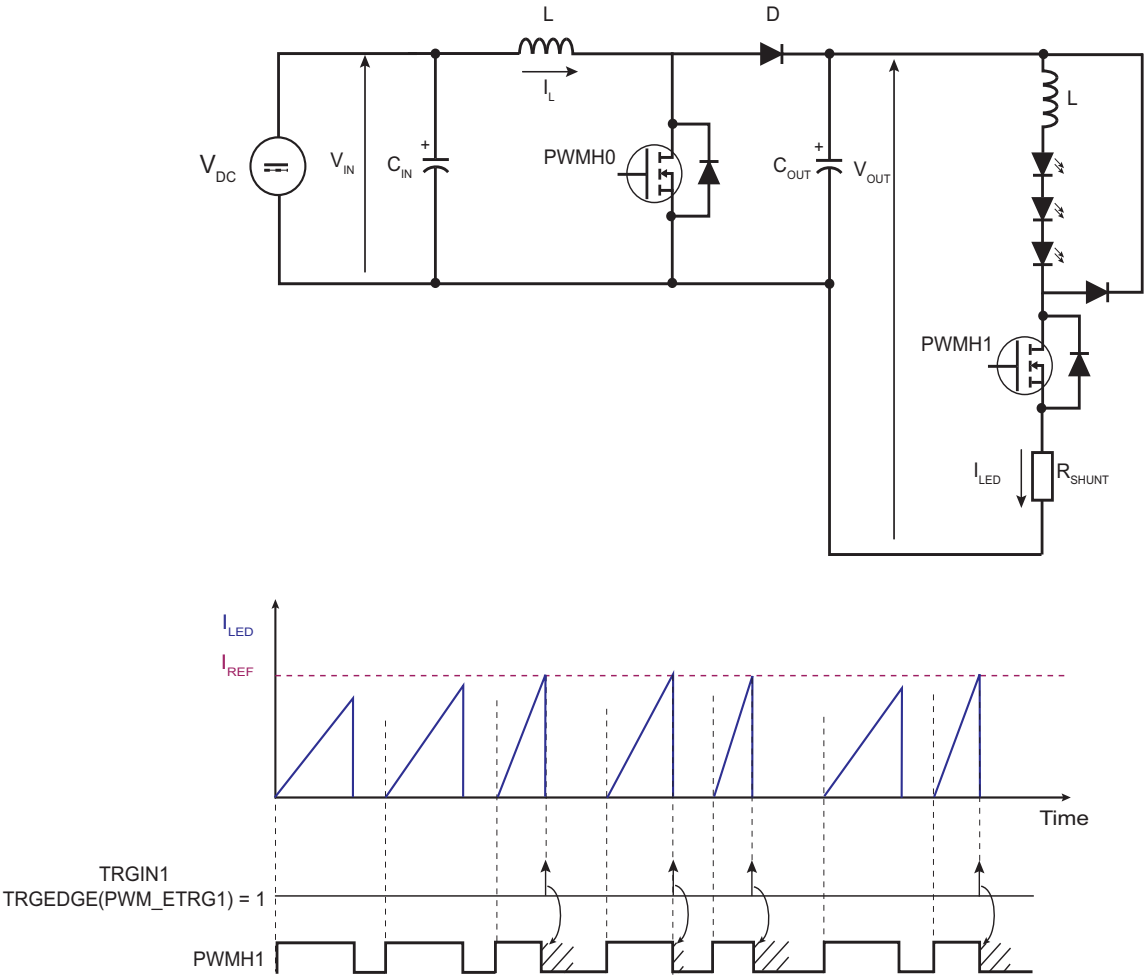
Downloaded from Arrow.com.

Application Example

Figure 49-32 illustrates an application example of the Cycle-by-cycle Duty mode.

In an LED string control circuit, Cycle-by-cycle Duty mode can be used to automatically limit the current in the LED string.

Figure 49-32. Cycle-By-Cycle Duty Mode: LED String Control



Downloaded from Arrow.com.

49.6.5.4 Leading-Edge Blanking (LEB)

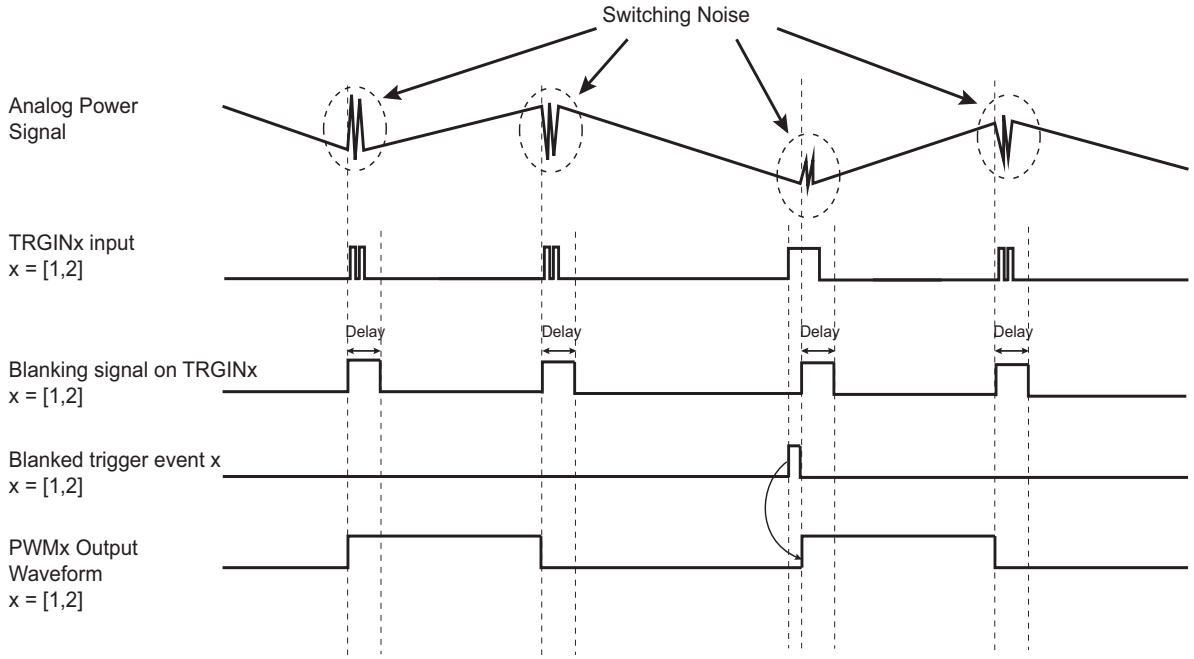
PWM channels 1 and 2 support leading-edge blanking. Leading-edge blanking masks the external trigger input when a transient occurs on the corresponding PWM output. It masks potential spurious external events due to power transistor switching.

The blanking delay on each external trigger input is configured by programming the LEBDELAYx in the PWM Leading-Edge Blanking Register.

The LEB can be enabled on both the rising and the falling edges for the PWMH and PWML outputs through the bits PWMLFEN, PWMLREN, PWMHFEN, PWMHREN.

Any event on the PWMTRGx input which occurs during the blanking time is ignored.

Figure 49-33. Leading-Edge Blanking



49.6.6 PWM Controller Operations

49.6.6.1 Initialization

Before enabling the channels, they must be configured by the software application as described below:

- Unlock User Interface by writing the WPCMD field in PWM_WPCR.
- Configuration of the clock generator (DIVA, PREA, DIVB, PREB in the PWM_CLK register if required).
- Selection of the clock for each channel (CPRE field in PWM_CMRx)
- Configuration of the waveform alignment for each channel (CALG field in PWM_CMRx)
- Selection of the counter event selection (if CALG = 1) for each channel (CES field in PWM_CMRx)
- Configuration of the output waveform polarity for each channel (CPOL bit in PWM_CMRx)
- Configuration of the period for each channel (CPRD in the PWM_CPRDx register). Writing in PWM_CPRDx register is possible while the channel is disabled. After validation of the channel, the user must use PWM_CPRDUPDx register to update PWM_CPRDx as explained below.
- Configuration of the duty-cycle for each channel (CDTY in the PWM_CDTYx register). Writing in PWM_CDTYx register is possible while the channel is disabled. After validation of the channel, the user must use PWM_CDTYUPDx register to update PWM_CDTYx as explained below.

Downloaded from Arrow.com.

- Configuration of the dead-time generator for each channel (DTH and DTL in PWM_DTx) if enabled (DTE bit in PWM_CMRx). Writing in the PWM_DTx register is possible while the channel is disabled. After validation of the channel, the user must use PWM_DTUPDx register to update PWM_DTx
- Selection of the synchronous channels (SYNCx in the PWM_SCM register)
- Selection of the moment when the WRDY flag and the corresponding DMA Controller transfer request are set (PTRM and PTRCS in the PWM_SCM register)
- Configuration of the Update mode (UPDM in PWM_SCM register)
- Configuration of the update period (UPR in PWM_SCUP register) if needed
- Configuration of the comparisons (PWM_CMPVx and PWM_CMPMx)
- Configuration of the event lines (PWM_ELMRx)
- Configuration of the fault inputs polarity (FPOL in PWM_FMR)
- Configuration of the fault protection (FMOD and FFIL in PWM_FMR, PWM_FPV and PWM_FPE1)
- Enable of the interrupts (writing CHIDx and FCHIDx in PWM_IER1, and writing WRDY, UNRE, CMPMx and CMPUx in PWM_IER2)
- Enable of the PWM channels (writing CHIDx in the PWM_ENA register)

49.6.6.2 Source Clock Selection Criteria

The large number of source clocks can make selection difficult. The relationship between the value in the PWM Channel Period Register (PWM_CPRDx) and the PWM Channel Duty Cycle Register (PWM_CDTYx) helps the user select the appropriate clock. The event number written in the Period Register gives the PWM accuracy. The Duty-Cycle quantum cannot be lower than $1/CPRDx$ value. The higher the value of PWM_CPRDx, the greater the PWM accuracy.

For example, if the user sets 15 (in decimal) in PWM_CPRDx, the user is able to set a value from between 1 up to 14 in PWM_CDTYx. The resulting duty-cycle quantum cannot be lower than 1/15 of the PWM period.

49.6.6.3 Changing the Duty-Cycle, the Period and the Dead-Times

It is possible to modulate the output waveform duty-cycle, period and dead-times.

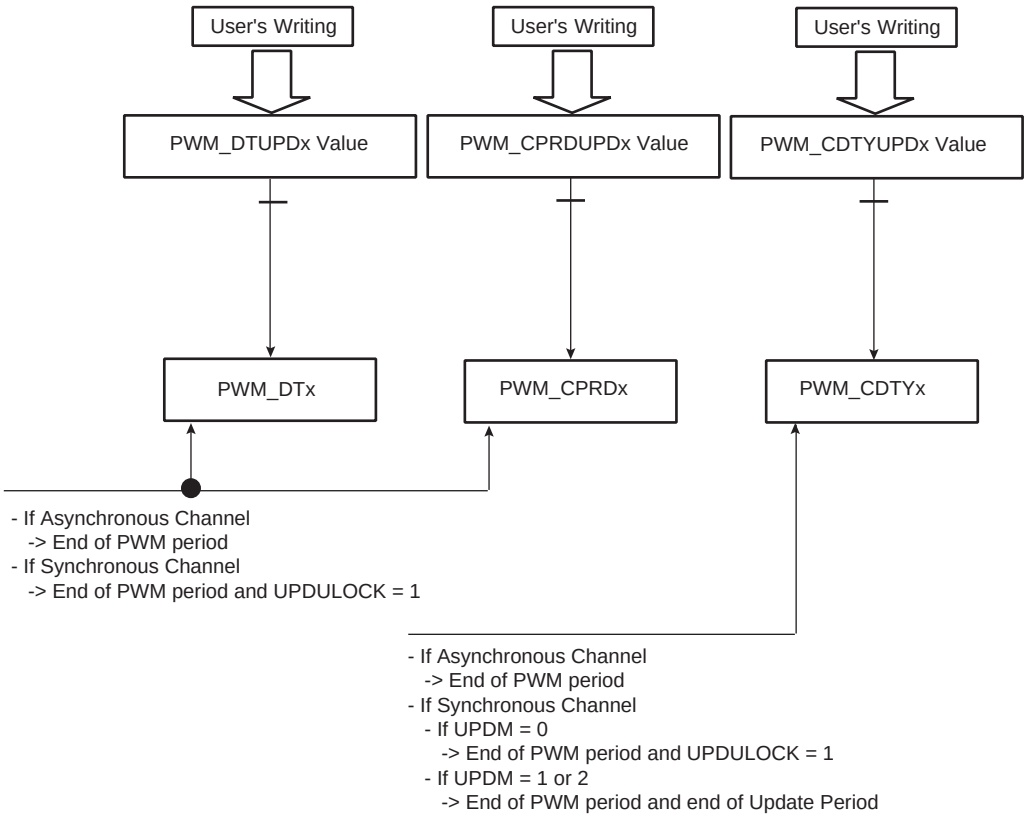
To prevent unexpected output waveform, the user must use the PWM Channel Duty Cycle Update Register (PWM_CDTYUPDx), the PWM Channel Period Update Register (PWM_CPRDUPDx) and the PWM Channel Dead Time Update Register (PWM_DTUPDx) to change waveform parameters while the channel is still enabled.

- If the channel is an asynchronous channel (SYNCx = 0 in PWM Sync Channels Mode Register (PWM_SCM)), these registers hold the new period, duty-cycle and dead-times values until the end of the current PWM period and update the values for the next period.
- If the channel is a synchronous channel and update method 0 is selected (SYNCx = 1 and UPDM = 0 in PWM_SCM register), these registers hold the new period, duty-cycle and dead-times values until the bit UPDULOCK is written at ‘1’ (in PWM Sync Channels Update Control Register (PWM_SCUC)) and the end of the current PWM period, then update the values for the next period.
- If the channel is a synchronous channel and update method 1 or 2 is selected (SYNCx = 1 and UPDM = 1 or 2 in PWM_SCM register):
  - registers PWM_CPRDUPDx and PWM_DTUPDx hold the new period and dead-times values until the bit UPDULOCK is written at ‘1’ (in PWM_SCUC) and the end of the current PWM period, then update the values for the next period.
  - register PWM_CDTYUPDx holds the new duty-cycle value until the end of the update period of synchronous channels (when UPRCNT is equal to UPR in PWM Sync Channels Update Period Register (PWM_SCUP)) and the end of the current PWM period, then updates the value for the next period.

Downloaded from Arrow.com.

Note: If the update registers PWM_CDTYUPDx, PWM_CPRDUPDx and PWM_DTUPDx are written several times between two updates, only the last written value is taken into account.

Figure 49-34. Synchronized Period, Duty-Cycle and Dead-Time Update



Downloaded from Arrow.com.

49.6.6.4 Changing the Update Period of Synchronous Channels

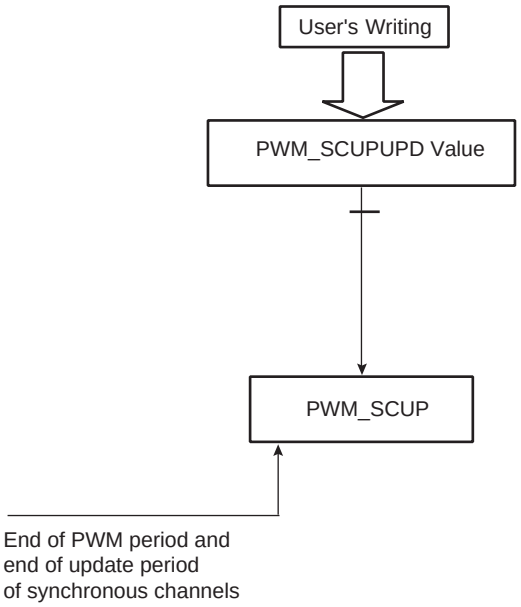
It is possible to change the update period of synchronous channels while they are enabled. See "Method 2: Manual write of duty-cycle values and automatic trigger of the update" and "Method 3: Automatic write of duty-cycle values and automatic trigger of the update" .

To prevent an unexpected update of the synchronous channels registers, the user must use the PWM Sync Channels Update Period Update Register (PWM_SCUPUPD) to change the update period of synchronous channels while they are still enabled. This register holds the new value until the end of the update period of synchronous channels (when UPRCNT is equal to UPR in PWM_SCUP) and the end of the current PWM period, then updates the value for the next period.

Note: If the update register PWM_SCUPUPD is written several times between two updates, only the last written value is taken into account.

Note: Changing the update period does make sense only if there is one or more synchronous channels and if the update method 1 or 2 is selected (UPDM = 1 or 2 in PWM Sync Channels Mode Register).

Figure 49-35. Synchronized Update of Update Period Value of Synchronous Channels



Downloaded from Arrow.com.

49.6.6.5 Changing the Comparison Value and the Comparison Configuration

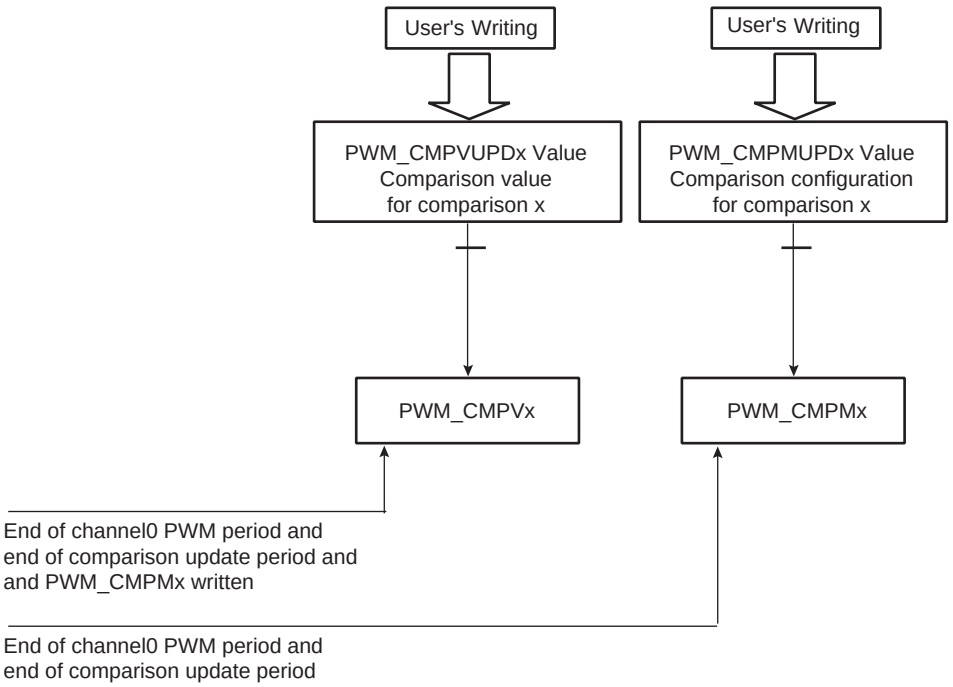
It is possible to change the comparison values and the comparison configurations while the channel 0 is enabled (see Section 49.6.3 “PWM Comparison Units”).

To prevent unexpected comparison match, the user must use the PWM Comparison x Value Update Register (PWM_CMPVUPDx) and the PWM Comparison x Mode Update Register (PWM_CMPMUPDx) to change, respectively, the comparison values and the comparison configurations while the channel 0 is still enabled. These registers hold the new values until the end of the comparison update period (when CUPRCNT is equal to CUPR in PWM Comparison x Mode Register (PWM_CMPMx) and the end of the current PWM period, then update the values for the next period.

CAUTION: The write of the register PWM_CMPVUPDx must be followed by a write of the register PWM_CMPMUPDx.

Note: If the update registers PWM_CMPVUPDx and PWM_CMPMUPDx are written several times between two updates, only the last written value are taken into account.

Figure 49-36. Synchronized Update of Comparison Values and Configurations



49.6.6.6 Interrupt Sources

Depending on the interrupt mask in PWM_IMR1 and PWM_IMR2, an interrupt can be generated at the end of the corresponding channel period (CHIDx in the PWM Interrupt Status Register 1 (PWM_ISR1)), after a fault event (FCHIDx in PWM_ISR1), after a comparison match (CMPMx in PWM_ISR2), after a comparison update (CMPUx in PWM_ISR2) or according to the Transfer mode of the synchronous channels (WRDY and UNRE in PWM_ISR2).

If the interrupt is generated by the flags CHIDx or FCHIDx, the interrupt remains active until a read operation in PWM_ISR1 occurs.

If the interrupt is generated by the flags WRDY or UNRE or CMPMx or CMPUx, the interrupt remains active until a read operation in PWM_ISR2 occurs.

A channel interrupt is enabled by setting the corresponding bit in PWM_IER1 and PWM_IER2. A channel interrupt is disabled by setting the corresponding bit in PWM_IDR1 and PWM_IDR2.

Downloaded from Arrow.com.

49.6.7 Register Write Protection

To prevent any single software error that may corrupt PWM behavior, the registers listed below can be write-protected by writing the field WPCMD in the PWM Write Protection Control Register (PWM_WPCR). They are divided into six groups:

- Register group 0:
  - PWM Clock Register
- Register group 1:
  - PWM Disable Register
- Register group 2:
  - PWM Sync Channels Mode Register
  - PWM Channel Mode Register
  - PWM Stepper Motor Mode Register
  - PWM Fault Protection Value Register 2
  - PWM Leading-Edge Blanking Register
  - PWM Channel Mode Update Register
- Register group 3:
  - PWM Spread Spectrum Register
  - PWM Spread Spectrum Update Register
  - PWM Channel Period Register
  - PWM Channel Period Update Register
- Register group 4:
  - PWM Channel Dead Time Register
  - PWM Channel Dead Time Update Register
- Register group 5:
  - PWM Fault Mode Register
  - PWM Fault Protection Value Register 1

There are two types of write protection:

- SW write protection—can be enabled or disabled by software
- HW write protection—can be enabled by software but only disabled by a hardware reset of the PWM controller

Both types of write protection can be applied independently to a particular register group by means of the WPCMD and WPRGx fields in PWM_WPCR. If at least one type of write protection is active, the register group is write-protected. The value of field WPCMD defines the action to be performed:

- 0: Disables SW write protection of the register groups of which the bit WPRGx is at ‘1’
- 1: Enables SW write protection of the register groups of which the bit WPRGx is at ‘1’
- 2: Enables HW write protection of the register groups of which the bit WPRGx is at ‘1’

At any time, the user can determine whether SW or HW write protection is active in a particular register group by the fields WPSWS and WPHWS in the PWM Write Protection Status Register (PWM_WPSR).

If a write access to a write-protected register is detected, the WPVS flag in PWM_WPSR is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS and WPVSRC fields are automatically cleared after reading PWM_WPSR.

Downloaded from Arrow.com.

49.7 Pulse Width Modulation Controller (PWM) User Interface

Table 49-8. Register Mapping

Offset	Register	Name	Access	Reset
0x00	PWM Clock Register	PWM_CLK	Read/Write	0x0
0x04	PWM Enable Register	PWM_ENA	Write-only	–
0x08	PWM Disable Register	PWM_DIS	Write-only	–
0x0C	PWM Status Register	PWM_SR	Read-only	0x0
0x10	PWM Interrupt Enable Register 1	PWM_IER1	Write-only	–
0x14	PWM Interrupt Disable Register 1	PWM_IDR1	Write-only	–
0x18	PWM Interrupt Mask Register 1	PWM_IMR1	Read-only	0x0
0x1C	PWM Interrupt Status Register 1	PWM_ISR1	Read-only	0x0
0x20	PWM Sync Channels Mode Register	PWM_SCM	Read/Write	0x0
0x24	PWM DMA Register	PWM_DMAR	Write-only	–
0x28	PWM Sync Channels Update Control Register	PWM_SCUC	Read/Write	0x0
0x2C	PWM Sync Channels Update Period Register	PWM_SCUP	Read/Write	0x0
0x30	PWM Sync Channels Update Period Update Register	PWM_SCUPUPD	Write-only	–
0x34	PWM Interrupt Enable Register 2	PWM_IER2	Write-only	–
0x38	PWM Interrupt Disable Register 2	PWM_IDR2	Write-only	–
0x3C	PWM Interrupt Mask Register 2	PWM_IMR2	Read-only	0x0
0x40	PWM Interrupt Status Register 2	PWM_ISR2	Read-only	0x0
0x44	PWM Output Override Value Register	PWM_OOV	Read/Write	0x0
0x48	PWM Output Selection Register	PWM_OS	Read/Write	0x0
0x4C	PWM Output Selection Set Register	PWM_OSS	Write-only	–
0x50	PWM Output Selection Clear Register	PWM_OSC	Write-only	–
0x54	PWM Output Selection Set Update Register	PWM_OSSUPD	Write-only	–
0x58	PWM Output Selection Clear Update Register	PWM_OSCUPD	Write-only	–
0x5C	PWM Fault Mode Register	PWM_FMR	Read/Write	0x0
0x60	PWM Fault Status Register	PWM_FSR	Read-only	0x0
0x64	PWM Fault Clear Register	PWM_FCR	Write-only	–
0x68	PWM Fault Protection Value Register 1	PWM_FPV1	Read/Write	0x0
0x6C	PWM Fault Protection Enable Register	PWM_FPE	Read/Write	0x0
0x70–0x78	Reserved	–	–	–
0x7C	PWM Event Line 0 Mode Register	PWM_ELMR0	Read/Write	0x0
0x80	PWM Event Line 1 Mode Register	PWM_ELMR1	Read/Write	0x0
0x84–0x9C	Reserved	–	–	–
0xA0	PWM Spread Spectrum Register	PWM_SSPR	Read/Write	0x0
0xA4	PWM Spread Spectrum Update Register	PWM_SSPUP	Write-only	–
0xA8–0xAC	Reserved	–	–	–

Downloaded from Arrow.com.

Table 49-8. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0xB0	PWM Stepper Motor Mode Register	PWM_SMMR	Read/Write	0x0
0xC0	PWM Fault Protection Value 2 Register	PWM_FPV2	Read/Write	0x003F_003F
0xC4–0xE0	Reserved	–	–	–
0xE4	PWM Write Protection Control Register	PWM_WPCR	Write-only	–
0xE8	PWM Write Protection Status Register	PWM_WPSR	Read-only	0x0
0xEC–0xFC	Reserved	–	–	–
0x100–0x12C	Reserved	–	–	–
0x130	PWM Comparison 0 Value Register	PWM_CMPV0	Read/Write	0x0
0x134	PWM Comparison 0 Value Update Register	PWM_CMPVUPD0	Write-only	–
0x138	PWM Comparison 0 Mode Register	PWM_CMPM0	Read/Write	0x0
0x13C	PWM Comparison 0 Mode Update Register	PWM_CMPMUPD0	Write-only	–
0x140	PWM Comparison 1 Value Register	PWM_CMPV1	Read/Write	0x0
0x144	PWM Comparison 1 Value Update Register	PWM_CMPVUPD1	Write-only	–
0x148	PWM Comparison 1 Mode Register	PWM_CMPM1	Read/Write	0x0
0x14C	PWM Comparison 1 Mode Update Register	PWM_CMPMUPD1	Write-only	–
0x150	PWM Comparison 2 Value Register	PWM_CMPV2	Read/Write	0x0
0x154	PWM Comparison 2 Value Update Register	PWM_CMPVUPD2	Write-only	–
0x158	PWM Comparison 2 Mode Register	PWM_CMPM2	Read/Write	0x0
0x15C	PWM Comparison 2 Mode Update Register	PWM_CMPMUPD2	Write-only	–
0x160	PWM Comparison 3 Value Register	PWM_CMPV3	Read/Write	0x0
0x164	PWM Comparison 3 Value Update Register	PWM_CMPVUPD3	Write-only	–
0x168	PWM Comparison 3 Mode Register	PWM_CMPM3	Read/Write	0x0
0x16C	PWM Comparison 3 Mode Update Register	PWM_CMPMUPD3	Write-only	–
0x170	PWM Comparison 4 Value Register	PWM_CMPV4	Read/Write	0x0
0x174	PWM Comparison 4 Value Update Register	PWM_CMPVUPD4	Write-only	–
0x178	PWM Comparison 4 Mode Register	PWM_CMPM4	Read/Write	0x0
0x17C	PWM Comparison 4 Mode Update Register	PWM_CMPMUPD4	Write-only	–
0x180	PWM Comparison 5 Value Register	PWM_CMPV5	Read/Write	0x0
0x184	PWM Comparison 5 Value Update Register	PWM_CMPVUPD5	Write-only	–
0x188	PWM Comparison 5 Mode Register	PWM_CMPM5	Read/Write	0x0
0x18C	PWM Comparison 5 Mode Update Register	PWM_CMPMUPD5	Write-only	–
0x190	PWM Comparison 6 Value Register	PWM_CMPV6	Read/Write	0x0
0x194	PWM Comparison 6 Value Update Register	PWM_CMPVUPD6	Write-only	–
0x198	PWM Comparison 6 Mode Register	PWM_CMPM6	Read/Write	0x0
0x19C	PWM Comparison 6 Mode Update Register	PWM_CMPMUPD6	Write-only	–
0x1A0	PWM Comparison 7 Value Register	PWM_CMPV7	Read/Write	0x0
0x1A4	PWM Comparison 7 Value Update Register	PWM_CMPVUPD7	Write-only	–

Downloaded from Arrow.com.

Table 49-8. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x1A8	PWM Comparison 7 Mode Register	PWM_CMPM7	Read/Write	0x0
0x1AC	PWM Comparison 7 Mode Update Register	PWM_CMPMUPD7	Write-only	–
0x1B0–0x1FC	Reserved	–	–	–
0x200 + ch_num * 0x20 + 0x00	PWM Channel Mode Register(1)	PWM_CMR	Read/Write	0x0
0x200 + ch_num * 0x20 + 0x04	PWM Channel Duty Cycle Register(1)	PWM_CDTY	Read/Write	0x0
0x200 + ch_num * 0x20 + 0x08	PWM Channel Duty Cycle Update Register(1)	PWM_CDTYUPD	Write-only	–
0x200 + ch_num * 0x20 + 0x0C	PWM Channel Period Register(1)	PWM_CPRD	Read/Write	0x0
0x200 + ch_num * 0x20 + 0x10	PWM Channel Period Update Register(1)	PWM_CPRDUPD	Write-only	–
0x200 + ch_num * 0x20 + 0x14	PWM Channel Counter Register(1)	PWM_CCNT	Read-only	0x0
0x200 + ch_num * 0x20 + 0x18	PWM Channel Dead Time Register(1)	PWM_DT	Read/Write	0x0
0x200 + ch_num * 0x20 + 0x1C	PWM Channel Dead Time Update Register(1)	PWM_DTUPD	Write-only	–
0x400 + ch_num * 0x20 + 0x00	PWM Channel Mode Update Register(1)	PWM_CMUPD	Write-only	–
0x400 + trg_num * 0x20 + 0x0C	PWM External Trigger Register(2)	PWM_ETRG	Read/Write	0x0
0x400 + trg_num * 0x20 + 0x10	PWM Leading-Edge Blanking Register(2)	PWM_LEBR	Read/Write	0x0

Notes: 1. Some registers are indexed with “ch_num” index ranging from 0 to 3.
2. Some registers are indexed with “trg_num” index ranging from 1 to 2.

Downloaded from Arrow.com.

49.7.1 PWM Clock Register

Name: PWM_CLK

Address: 0x40020000 (0), 0x4005C000 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	PREB			
23	22	21	20	19	18	17	16
DIVB							
15	14	13	12	11	10	9	8
–	–	–	–	PREA			
7	6	5	4	3	2	1	0
DIVA							

This register can only be written if bits WPSWS0 and WPHWS0 are cleared in the PWM Write Protection Status Register.

- **DIVA: CLKA Divide Factor**

Value	Name	Description
0	CLKA_POFF	CLKA clock is turned off
1	PREA	CLKA clock is clock selected by PREA
2–255	PREA_DIV	CLKA clock is clock selected by PREA divided by DIVA factor

- **DIVB: CLKB Divide Factor**

Value	Name	Description
0	CLKB_POFF	CLKB clock is turned off
1	PREB	CLKB clock is clock selected by PREB
2–255	PREB_DIV	CLKB clock is clock selected by PREB divided by DIVB factor

- **PREA: CLKA Source Clock Selection**

Value	Name	Description
0	CLK	Peripheral clock
1	CLK_DIV2	Peripheral clock/2
2	CLK_DIV4	Peripheral clock/4
3	CLK_DIV8	Peripheral clock/8
4	CLK_DIV16	Peripheral clock/16
5	CLK_DIV32	Peripheral clock/32
6	CLK_DIV64	Peripheral clock/64
7	CLK_DIV128	Peripheral clock/128
8	CLK_DIV256	Peripheral clock/256

Downloaded from Arrow.com.

9	CLK_DIV512	Peripheral clock/512
10	CLK_DIV1024	Peripheral clock/1024
Other	–	Reserved

- **PREB: CLKB Source Clock Selection**

Value	Name	Description
0	CLK	Peripheral clock
1	CLK_DIV2	Peripheral clock/2
2	CLK_DIV4	Peripheral clock/4
3	CLK_DIV8	Peripheral clock/8
4	CLK_DIV16	Peripheral clock/16
5	CLK_DIV32	Peripheral clock/32
6	CLK_DIV64	Peripheral clock/64
7	CLK_DIV128	Peripheral clock/128
8	CLK_DIV256	Peripheral clock/256
9	CLK_DIV512	Peripheral clock/512
10	CLK_DIV1024	Peripheral clock/1024
Other	–	Reserved

Downloaded from Arrow.com.

49.7.2 PWM Enable Register

Name: PWM_ENA

Address: 0x40020004 (0), 0x4005C004 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**

0: No effect.

1: Enable PWM output for channel x.

Downloaded from Arrow.com.

49.7.3 PWM Disable Register

Name: PWM_DIS

Address: 0x40020008 (0), 0x4005C008 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

This register can only be written if bits WPSWS1 and WPHWS1 are cleared in the PWM Write Protection Status Register.

- **CHIDx: Channel ID**

0: No effect.

1: Disable PWM output for channel x.

Downloaded from Arrow.com.

49.7.4 PWM Status Register

Name: PWM_SR

Address: 0x4002000C (0), 0x4005C00C (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**

0: PWM output for channel x is disabled.

1: PWM output for channel x is enabled.

Downloaded from Arrow.com.

49.7.5 PWM Interrupt Enable Register 1

Name: PWM_IER1

Address: 0x40020010 (0), 0x4005C010 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	FCHID3	FCHID2	FCHID1	FCHID0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Counter Event on Channel x Interrupt Enable**
- **FCHIDx: Fault Protection Trigger on Channel x Interrupt Enable**

Downloaded from Arrow.com.

49.7.6 PWM Interrupt Disable Register 1

Name: PWM_IDR1

Address: 0x40020014 (0), 0x4005C014 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	FCHID3	FCHID2	FCHID1	FCHID0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Counter Event on Channel x Interrupt Disable**
- **FCHIDx: Fault Protection Trigger on Channel x Interrupt Disable**

Downloaded from Arrow.com.

49.7.7 PWM Interrupt Mask Register 1

Name: PWM_IMR1

Address: 0x40020018 (0), 0x4005C018 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	FCHID3	FCHID2	FCHID1	FCHID0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Counter Event on Channel x Interrupt Mask**
- **FCHIDx: Fault Protection Trigger on Channel x Interrupt Mask**

Downloaded from Arrow.com.

49.7.8 PWM Interrupt Status Register 1

Name: PWM_ISR1

Address: 0x4002001C (0), 0x4005C01C (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	FCHID3	FCHID2	FCHID1	FCHID0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Counter Event on Channel x**

0: No new counter event has occurred since the last read of PWM_ISR1.

1: At least one counter event has occurred since the last read of PWM_ISR1.

- **FCHIDx: Fault Protection Trigger on Channel x**

0: No new trigger of the fault protection since the last read of PWM_ISR1.

1: At least one trigger of the fault protection since the last read of PWM_ISR1.

Note: Reading PWM_ISR1 automatically clears CHIDx and FCHIDx flags.

Downloaded from Arrow.com.

49.7.9 PWM Sync Channels Mode Register

Name: PWM_SCM

Address: 0x40020020 (0), 0x4005C020 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
PTRCS			PTRM	–	–	UPDM	
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	SYNC3	SYNC2	SYNC1	SYNC0

This register can only be written if bits WPSWS2 and WPHWS2 are cleared in the PWM Write Protection Status Register.

- **SYNCx: Synchronous Channel x**

0: Channel x is not a synchronous channel.

1: Channel x is a synchronous channel.

- **UPDM: Synchronous Channels Update Mode**

Value	Name	Description
0	MODE0	Manual write of double buffer registers and manual update of synchronous channels(1)
1	MODE1	Manual write of double buffer registers and automatic update of synchronous channels(2)
2	MODE2	Automatic write of duty-cycle update registers by the DMA Controller and automatic update of synchronous channels(2)

Notes: 1. The update occurs at the beginning of the next PWM period, when the UPDULOCK bit in PWM Sync Channels Update Control Register is set.

2. The update occurs when the Update Period is elapsed.

- **PTRM: DMA Controller Transfer Request Mode**

UPDM	PTRM	WRDY Flag and DMA Controller Transfer Request
0	x	The WRDY flag in PWM Interrupt Status Register 2 and the DMA transfer request are never set to ‘1’.
1	x	The WRDY flag in PWM Interrupt Status Register 2 is set to ‘1’ as soon as the update period is elapsed, the DMA Controller transfer request is never set to ‘1’.
2	0	The WRDY flag in PWM Interrupt Status Register 2 and the DMA transfer request are set to ‘1’ as soon as the update period is elapsed.
	1	The WRDY flag in PWM Interrupt Status Register 2 and the DMA transfer request are set to ‘1’ as soon as the selected comparison matches.

- **PTRCS: DMA Controller Transfer Request Comparison Selection**

Selection of the comparison used to set the flag WRDY and the corresponding DMA Controller transfer request.

Downloaded from Arrow.com.

49.7.10 PWM DMA Register

Name: PWM_DMAR

Address: 0x40020024 (0), 0x4005C024 (1)

Access: Write- only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
DMADUTY							

15	14	13	12	11	10	9	8
DMADUTY							

7	6	5	4	3	2	1	0
DMADUTY							

Only the first 16 bits (channel counter size) are significant.

- **DMADUTY: Duty-Cycle Holding Register for DMA Access**

Each write access to PWM_DMAR sequentially updates the CDTY field of PWM_CDTYx with DMADUTY (only for channel configured as synchronous). See “Method 3: Automatic write of duty-cycle values and automatic trigger of the update” .

Downloaded from Arrow.com.

49.7.11 PWM Sync Channels Update Control Register

Name: PWM_SCUC

Address: 0x40020028 (0), 0x4005C028 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	UPDULOCK

- **UPDULOCK: Synchronous Channels Update Unlock**

0: No effect

1: If the UPDM field is set to ‘0’ in PWM Sync Channels Mode Register, writing the UPDULOCK bit to ‘1’ triggers the update of the period value, the duty-cycle and the dead-time values of synchronous channels at the beginning of the next PWM period. If the field UPDM is set to ‘1’ or ‘2’, writing the UPDULOCK bit to ‘1’ triggers only the update of the period value and of the dead-time values of synchronous channels.

This bit is automatically reset when the update is done.

Downloaded from Arrow.com.

49.7.12 PWM Sync Channels Update Period Register

Name: PWM_SCUP

Address: 0x4002002C (0), 0x4005C02C (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
UPRCNT				UPR			

- **UPR: Update Period**

Defines the time between each update of the synchronous channels if automatic trigger of the update is activated (UPDM = 1 or UPDM = 2 in PWM Sync Channels Mode Register). This time is equal to UPR+1 periods of the synchronous channels.

- **UPRCNT: Update Period Counter**

Reports the value of the update period counter.

Downloaded from Arrow.com.

49.7.13 PWM Sync Channels Update Period Update Register

Name: PWM_SCUPUPD

Address: 0x40020030 (0), 0x4005C030 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	UPRUPD			

This register acts as a double buffer for the UPR value. This prevents an unexpected automatic trigger of the update of synchronous channels.

- **UPRUPD: Update Period Update**

Defines the wanted time between each update of the synchronous channels if automatic trigger of the update is activated (UPDM = 1 or UPDM = 2 in PWM Sync Channels Mode Register). This time is equal to UPR+1 periods of the synchronous channels.

Downloaded from Arrow.com.

49.7.14 PWM Interrupt Enable Register 2

Name: PWM_IER2

Address: 0x40020034 (0), 0x4005C034 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
CMPU7	CMPU6	CMPU5	CMPU4	CMPU3	CMPU2	CMPU1	CMPU0
15	14	13	12	11	10	9	8
CMPM7	CMPM6	CMPM5	CMPM4	CMPM3	CMPM2	CMPM1	CMPM0
7	6	5	4	3	2	1	0
–	–	–	–	UNRE	–	–	WRDY

- **WRDY: Write Ready for Synchronous Channels Update Interrupt Enable**
- **UNRE: Synchronous Channels Update Underrun Error Interrupt Enable**
- **CMPMx: Comparison x Match Interrupt Enable**
- **CMPUx: Comparison x Update Interrupt Enable**

Downloaded from Arrow.com.

49.7.15 PWM Interrupt Disable Register 2

Name: PWM_IDR2

Address: 0x40020038 (0), 0x4005C038 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
CMPU7	CMPU6	CMPU5	CMPU4	CMPU3	CMPU2	CMPU1	CMPU0
15	14	13	12	11	10	9	8
CMPM7	CMPM6	CMPM5	CMPM4	CMPM3	CMPM2	CMPM1	CMPM0
7	6	5	4	3	2	1	0
–	–	–	–	UNRE	–	–	WRDY

- **WRDY: Write Ready for Synchronous Channels Update Interrupt Disable**
- **UNRE: Synchronous Channels Update Underrun Error Interrupt Disable**
- **CMPMx: Comparison x Match Interrupt Disable**
- **CMPUx: Comparison x Update Interrupt Disable**

Downloaded from Arrow.com.

49.7.16 PWM Interrupt Mask Register 2

Name: PWM_IMR2

Address: 0x4002003C (0), 0x4005C03C (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
CMPU7	CMPU6	CMPU5	CMPU4	CMPU3	CMPU2	CMPU1	CMPU0
15	14	13	12	11	10	9	8
CMPM7	CMPM6	CMPM5	CMPM4	CMPM3	CMPM2	CMPM1	CMPM0
7	6	5	4	3	2	1	0
–	–	–	–	UNRE	–	–	WRDY

- **WRDY: Write Ready for Synchronous Channels Update Interrupt Mask**
- **UNRE: Synchronous Channels Update Underrun Error Interrupt Mask**
- **CMPMx: Comparison x Match Interrupt Mask**
- **CMPUx: Comparison x Update Interrupt Mask**

Downloaded from Arrow.com.

49.7.17 PWM Interrupt Status Register 2

Name: PWM_ISR2

Address: 0x40020040 (0), 0x4005C040 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
CMPU7	CMPU6	CMPU5	CMPU4	CMPU3	CMPU2	CMPU1	CMPU0

15	14	13	12	11	10	9	8
CMPM7	CMPM6	CMPM5	CMPM4	CMPM3	CMPM2	CMPM1	CMPM0

7	6	5	4	3	2	1	0
–	–	–	–	UNRE	–	–	WRDY

- **WRDY: Write Ready for Synchronous Channels Update**

0: New duty-cycle and dead-time values for the synchronous channels cannot be written.

1: New duty-cycle and dead-time values for the synchronous channels can be written.

- **UNRE: Synchronous Channels Update Underrun Error**

0: No Synchronous Channels Update Underrun has occurred since the last read of the PWM_ISR2 register.

1: At least one Synchronous Channels Update Underrun has occurred since the last read of the PWM_ISR2 register.

- **CMPMx: Comparison x Match**

0: The comparison x has not matched since the last read of the PWM_ISR2 register.

1: The comparison x has matched at least one time since the last read of the PWM_ISR2 register.

- **CMPUx: Comparison x Update**

0: The comparison x has not been updated since the last read of the PWM_ISR2 register.

1: The comparison x has been updated at least one time since the last read of the PWM_ISR2 register.

Note: Reading PWM_ISR2 automatically clears flags WRDY, UNRE and CMPSx.

Downloaded from Arrow.com.

49.7.18 PWM Output Override Value Register

Name: PWM_OOV

Address: 0x40020044 (0), 0x4005C044 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	OOVL3	OOVL2	OOVL1	OOVL0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	OOVH3	OOVH2	OOVH1	OOVH0

- **OOVHx: Output Override Value for PWMH output of the channel x**

0: Override value is 0 for PWMH output of channel x.

1: Override value is 1 for PWMH output of channel x.

- **OOVLx: Output Override Value for PWML output of the channel x**

0: Override value is 0 for PWML output of channel x.

1: Override value is 1 for PWML output of channel x.

Downloaded from Arrow.com.

49.7.19 PWM Output Selection Register

Name: PWM_OS

Address: 0x40020048 (0), 0x4005C048 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	OSL3	OSL2	OSL1	OSL0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	OSH3	OSH2	OSH1	OSH0

- **OSHx: Output Selection for PWMH output of the channel x**

0: Dead-time generator output DTOHx selected as PWMH output of channel x.

1: Output override value OOVHx selected as PWMH output of channel x.

- **OSLx: Output Selection for PWML output of the channel x**

0: Dead-time generator output DTOLx selected as PWML output of channel x.

1: Output override value OOVLx selected as PWML output of channel x.

Downloaded from Arrow.com.

49.7.20 PWM Output Selection Set Register

Name: PWM_OSS

Address: 0x4002004C (0), 0x4005C04C (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	OSSL3	OSSL2	OSSL1	OSSL0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	OSSH3	OSSH2	OSSH1	OSSH0

- **OSSHx: Output Selection Set for PWMH output of the channel x**

0: No effect.

1: Output override value OOVHx selected as PWMH output of channel x.

- **OSSLx: Output Selection Set for PWML output of the channel x**

0: No effect.

1: Output override value OOVLx selected as PWML output of channel x.

Downloaded from Arrow.com.

49.7.21 PWM Output Selection Clear Register

Name: PWM_OSC

Address: 0x40020050 (0), 0x4005C050 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	OSCL3	OSCL2	OSCL1	OSCL0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	OSCH3	OSCH2	OSCH1	OSCH0

- **OSCHx: Output Selection Clear for PWMH output of the channel x**

0: No effect.

1: Dead-time generator output DTOHx selected as PWMH output of channel x.

- **OSCLx: Output Selection Clear for PWML output of the channel x**

0: No effect.

1: Dead-time generator output DTOLx selected as PWML output of channel x.

Downloaded from Arrow.com.

49.7.22 PWM Output Selection Set Update Register

Name: PWM_OSSUPD

Address: 0x40020054 (0), 0x4005C054 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	OSSUPL3	OSSUPL2	OSSUPL1	OSSUPL0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	OSSUPH3	OSSUPH2	OSSUPH1	OSSUPH0

- **OSSUPHx: Output Selection Set for PWMH output of the channel x**

0: No effect.

1: Output override value OOVHx selected as PWMH output of channel x at the beginning of the next channel x PWM period.

- **OSSUPLx: Output Selection Set for PWML output of the channel x**

0: No effect.

1: Output override value OOVLx selected as PWML output of channel x at the beginning of the next channel x PWM period.

Downloaded from Arrow.com.

49.7.23 PWM Output Selection Clear Update Register

Name: PWM_OSCUPD

Address: 0x40020058 (0), 0x4005C058 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	OSCUPL3	OSCUPL2	OSCUPL1	OSCUPL0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	OSCUPH3	OSCUPH2	OSCUPH1	OSCUPH0

- **OSCUPHx: Output Selection Clear for PWMH output of the channel x**

0: No effect.

1: Dead-time generator output DTOHx selected as PWMH output of channel x at the beginning of the next channel x PWM period.

- **OSCUPLx: Output Selection Clear for PWML output of the channel x**

0: No effect.

1: Dead-time generator output DTOLx selected as PWML output of channel x at the beginning of the next channel x PWM period.

Downloaded from Arrow.com.

49.7.24 PWM Fault Mode Register

Name: PWM_FMR

Address: 0x4002005C (0), 0x4005C05C (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–							
23	22	21	20	19	18	17	16
FFIL							
15	14	13	12	11	10	9	8
FMOD							
7	6	5	4	3	2	1	0
FPOL							

This register can only be written if bits WPSWS5 and WPHWS5 are cleared in the PWM Write Protection Status Register.

Refer to Section 49.5.4 “Fault Inputs” for details on fault generation.

- **FPOL: Fault Polarity**

For each bit y of FPOL, where y is the fault input number:

0: The fault y becomes active when the fault input y is at 0.

1: The fault y becomes active when the fault input y is at 1.

- **FMOD: Fault Activation Mode**

For each bit y of FMOD, where y is the fault input number:

0: The fault y is active until the fault condition is removed at the peripheral(1) level.

1: The fault y stays active until the fault condition is removed at the peripheral(1) level AND until it is cleared in the PWM Fault Clear Register.

Note: 1. The peripheral generating the fault.

- **FFIL: Fault Filtering**

For each bit y of FFIL, where y is the fault input number:

0: The fault input y is not filtered.

1: The fault input y is filtered.

CAUTION: To prevent an unexpected activation of the status flag FSy in the PWM Fault Status Register, the bit FMODy can be set to ‘1’ only if the FPOLy bit has been previously configured to its final value.

Downloaded from Arrow.com.

49.7.25 PWM Fault Status Register

Name: PWM_FSR

Address: 0x40020060 (0), 0x4005C060 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–							

23	22	21	20	19	18	17	16
–							

15	14	13	12	11	10	9	8
FS							

7	6	5	4	3	2	1	0
FIV							

Refer to Section 49.5.4 “Fault Inputs” for details on fault generation.

- **FIV: Fault Input Value**

For each bit y of FIV, where y is the fault input number:

0: The current sampled value of the fault input y is 0 (after filtering if enabled).

1: The current sampled value of the fault input y is 1 (after filtering if enabled).

- **FS: Fault Status**

For each bit y of FS, where y is the fault input number:

0: The fault y is not currently active.

1: The fault y is currently active.

Downloaded from Arrow.com.

49.7.26 PWM Fault Clear Register

Name: PWM_FCR

Address: 0x40020064 (0), 0x4005C064 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–							

23	22	21	20	19	18	17	16
–							

15	14	13	12	11	10	9	8
–							

7	6	5	4	3	2	1	0
FCLR							

Refer to Section 49.5.4 “Fault Inputs” for details on fault generation.

- **FCLR: Fault Clear**

For each bit y of FCLR, where y is the fault input number:

0: No effect.

1: If bit y of FMOD field is set to ‘1’ and if the fault input y is not at the level defined by the bit y of FPOL field, the fault y is cleared and becomes inactive (FMOD and FPOL fields belong to PWM Fault Mode Register), else writing this bit to ‘1’ has no effect.

Downloaded from Arrow.com.

49.7.27 PWM Fault Protection Value Register 1

Name: PWM_FPV1

Address: 0x40020068 (0), 0x4005C068 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	FPVL3	FPVL2	FPVL1	FPVL0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	FPVH3	FPVH2	FPVH1	FPVH0

This register can only be written if bits WPSWS5 and WPHWS5 are cleared in the PWM Write Protection Status Register.

Refer to Section 49.5.4 “Fault Inputs” for details on fault generation.

- **FPVHx: Fault Protection Value for PWMH output on channel x**

This bit is taken into account only if the bit FPZHx is set to ‘0’ in PWM Fault Protection Value Register 2.

0: PWMH output of channel x is forced to ‘0’ when fault occurs.

1: PWMH output of channel x is forced to ‘1’ when fault occurs.

- **FPVLx: Fault Protection Value for PWML output on channel x**

This bit is taken into account only if the bit FPZLx is set to ‘0’ in PWM Fault Protection Value Register 2.

0: PWML output of channel x is forced to ‘0’ when fault occurs.

1: PWML output of channel x is forced to ‘1’ when fault occurs.

Downloaded from Arrow.com.

49.7.28 PWM Fault Protection Enable Register

Name: PWM_FPE

Address: 0x4002006C (0), 0x4005C06C (1)

Access: Read/Write

31	30	29	28	27	26	25	24
FPE3							
23	22	21	20	19	18	17	16
FPE2							
15	14	13	12	11	10	9	8
FPE1							
7	6	5	4	3	2	1	0
FPE0							

This register can only be written if bits WPSWS5 and WPHWS5 are cleared in the PWM Write Protection Status Register.

Only the first 8 bits (number of fault input pins) of fields FPE0, FPE1, FPE2 and FPE3 are significant.

Refer to Section 49.5.4 “Fault Inputs” for details on fault generation.

- **FPEx: Fault Protection Enable for channel x**

For each bit y of FPEx, where y is the fault input number:

0: Fault y is not used for the fault protection of channel x.

1: Fault y is used for the fault protection of channel x.

CAUTION: To prevent an unexpected activation of the fault protection, the bit y of FPEx field can be set to ‘1’ only if the corresponding FPOL field has been previously configured to its final value in PWM Fault Mode Register.

Downloaded from Arrow.com.

49.7.29 PWM Event Line x Register

Name: PWM_ELMRx

Address: 0x4002007C (0), 0x4005C07C (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
CSEL7	CSEL6	CSEL5	CSEL4	CSEL3	CSEL2	CSEL1	CSEL0

- **CSELy: Comparison y Selection**

0: A pulse is not generated on the event line x when the comparison y matches.

1: A pulse is generated on the event line x when the comparison y match.

Downloaded from Arrow.com.

49.7.30 PWM Spread Spectrum Register

Name: PWM_SSPR

Address: 0x400200A0 (0), 0x4005C0A0 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	SPRDM
23	22	21	20	19	18	17	16
SPRD							
15	14	13	12	11	10	9	8
SPRD							
7	6	5	4	3	2	1	0
SPRD							

This register can only be written if bits WPSWS3 and WPHWS3 are cleared in the PWM Write Protection Status Register.

Only the first 16 bits (channel counter size) are significant.

- **SPRD: Spread Spectrum Limit Value**

The spread spectrum limit value defines the range for the spread spectrum counter. It is introduced in order to achieve constant varying PWM period for the output waveform.

- **SPRDM: Spread Spectrum Counter Mode**

0: Triangular mode. The spread spectrum counter starts to count from -SPRD when the channel 0 is enabled and counts upwards at each PWM period. When it reaches +SPRD, it restarts to count from -SPRD again.

1: Random mode. The spread spectrum counter is loaded with a new random value at each PWM period. This random value is uniformly distributed and is between -SPRD and +SPRD.

Downloaded from Arrow.com.

49.7.31 PWM Spread Spectrum Update Register

Name: PWM_SSPUP

Address: 0x400200A4 (0), 0x4005C0A4 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
SPRDUP							

15	14	13	12	11	10	9	8
SPRDUP							

7	6	5	4	3	2	1	0
SPRDUP							

This register can only be written if bits WPSWS3 and WPHWS3 are cleared in the PWM Write Protection Status Register.

This register acts as a double buffer for the SPRD value. This prevents an unexpected waveform when modifying the spread spectrum limit value.

Only the first 16 bits (channel counter size) are significant.

- **SPRDUP: Spread Spectrum Limit Value Update**

The spread spectrum limit value defines the range for the spread spectrum counter. It is introduced in order to achieve constant varying period for the output waveform.

Downloaded from Arrow.com.

49.7.32 PWM Stepper Motor Mode Register

Name: PWM_SMMR

Address: 0x400200B0 (0), 0x4005C0B0 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	DOWN1	DOWN0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	GCEN1	GCEN0

- **GCENx: Gray Count ENable**

0: Disable gray count generation on PWML[2*x], PWMH[2*x], PWML[2*x +1], PWMH[2*x +1]

1: Enable gray count generation on PWML[2*x], PWMH[2*x], PWML[2*x +1], PWMH[2*x +1.

- **DOWNx: DOWN Count**

0: Up counter.

1: Down counter.

Downloaded from Arrow.com.

49.7.33 PWM Fault Protection Value Register 2

Name: PWM_FPV2

Address: 0x400200C0 (0), 0x4005C0C0 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	FPZL3	FPZL2	FPZL1	FPZL0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	FPZH3	FPZH2	FPZH1	FPZH0

This register can only be written if bits WPSWS5 and WPHWS5 are cleared in the PWM Write Protection Status Register.

- **FPZHx: Fault Protection to Hi-Z for PWMH output on channel x**

0: When fault occurs, PWMH output of channel x is forced to value defined by the bit FPVHx in PWM Fault Protection Value Register 1.

1: When fault occurs, PWMH output of channel x is forced to high-impedance state.

- **FPZLx: Fault Protection to Hi-Z for PWML output on channel x**

0: When fault occurs, PWML output of channel x is forced to value defined by the bit FPVLx in PWM Fault Protection Value Register 1.

1: When fault occurs, PWML output of channel x is forced to high-impedance state.

Downloaded from Arrow.com.

49.7.34 PWM Write Protection Control Register

Name: PWM_WPCR

Address: 0x400200E4 (0), 0x4005C0E4 (1)

Access: Write-only

31	30	29	28	27	26	25	24
WPKEY							

23	22	21	20	19	18	17	16
WPKEY							

15	14	13	12	11	10	9	8
WPKEY							

7	6	5	4	3	2	1	0
WPRG5	WPRG4	WPRG3	WPRG2	WPRG1	WPRG0	WPCMD	

See Section 49.6.7 “Register Write Protection” for the list of registers that can be write-protected.

- **WPCMD: Write Protection Command**

This command is performed only if the WPKEY corresponds to 0x50574D (“PWM” in ASCII).

Value	Name	Description
0	DISABLE_SW_PROT	Disables the software write protection of the register groups of which the bit WPRGx is at ‘1’.
1	ENABLE_SW_PROT	Enables the software write protection of the register groups of which the bit WPRGx is at ‘1’.
2	ENABLE_HW_PROT	Enables the hardware write protection of the register groups of which the bit WPRGx is at ‘1’. Only a hardware reset of the PWM controller can disable the hardware write protection. Moreover, to meet security requirements, the PIO lines associated with the PWM can not be configured through the PIO interface.

- **WPRGx: Write Protection Register Group x**

0: The WPCMD command has no effect on the register group x.

1: The WPCMD command is applied to the register group x.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x50574D	PASSWD	Writing any other value in this field aborts the write operation of the WPCMD field. Always reads as 0

Downloaded from Arrow.com.

49.7.35 PWM Write Protection Status Register

Name: PWM_WPSR

Address: 0x400200E8 (0), 0x4005C0E8 (1)

Access: Read-only

31	30	29	28	27	26	25	24
WPVSRC							

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
–	–	WPHWS5	WPHWS4	WPHWS3	WPHWS2	WPHWS1	WPHWS0

7	6	5	4	3	2	1	0
WPVS	–	WPSWS5	WPSWS4	WPSWS3	WPSWS2	WPSWS1	WPSWS0

- **WPSWSx: Write Protect SW Status**

0: The SW write protection x of the register group x is disabled.

1: The SW write protection x of the register group x is enabled.

- **WPHWSx: Write Protect HW Status**

0: The HW write protection x of the register group x is disabled.

1: The HW write protection x of the register group x is enabled.

- **WPVS: Write Protect Violation Status**

0: No write protection violation has occurred since the last read of PWM_WPSR.

1: At least one write protection violation has occurred since the last read of PWM_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protect Violation Source**

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

Downloaded from Arrow.com.

49.7.36 PWM Comparison x Value Register

Name: PWM_CMPVx

Address: 0x40020130 (0)[0], 0x40020140 (0)[1], 0x40020150 (0)[2], 0x40020160 (0)[3], 0x40020170 (0)[4], 0x40020180 (0)[5], 0x40020190 (0)[6], 0x400201A0 (0)[7], 0x4005C130 (1)[0], 0x4005C140 (1)[1], 0x4005C150 (1)[2], 0x4005C160 (1)[3], 0x4005C170 (1)[4], 0x4005C180 (1)[5], 0x4005C190 (1)[6], 0x4005C1A0 (1)[7]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	CVM
23	22	21	20	19	18	17	16
CV							
15	14	13	12	11	10	9	8
CV							
7	6	5	4	3	2	1	0
CV							

Only the first 16 bits (channel counter size) of field CV are significant.

- **CV: Comparison x Value**

Define the comparison x value to be compared with the counter of the channel 0.

- **CVM: Comparison x Value Mode**

0: The comparison x between the counter of the channel 0 and the comparison x value is performed when this counter is incrementing.

1: The comparison x between the counter of the channel 0 and the comparison x value is performed when this counter is decrementing.

Note: This bit is not relevant if the counter of the channel 0 is left-aligned (CALG = 0 in PWM Channel Mode Register)

Downloaded from Arrow.com.

49.7.37 PWM Comparison x Value Update Register

Name: PWM_CMPVUPDx

Address: 0x40020134 (0)[0], 0x40020144 (0)[1], 0x40020154 (0)[2], 0x40020164 (0)[3], 0x40020174 (0)[4], 0x40020184 (0)[5], 0x40020194 (0)[6], 0x400201A4 (0)[7], 0x4005C134 (1)[0], 0x4005C144 (1)[1], 0x4005C154 (1)[2], 0x4005C164 (1)[3], 0x4005C174 (1)[4], 0x4005C184 (1)[5], 0x4005C194 (1)[6], 0x4005C1A4 (1)[7]

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	CVMUPD
23	22	21	20	19	18	17	16
CVUPD							
15	14	13	12	11	10	9	8
CVUPD							
7	6	5	4	3	2	1	0
CVUPD							

This register acts as a double buffer for the CV and CVM values. This prevents an unexpected comparison x match.

Only the first 16 bits (channel counter size) of field CVUPD are significant.

- **CVUPD: Comparison x Value Update**

Define the comparison x value to be compared with the counter of the channel 0.

- **CVMUPD: Comparison x Value Mode Update**

0: The comparison x between the counter of the channel 0 and the comparison x value is performed when this counter is incrementing.

1: The comparison x between the counter of the channel 0 and the comparison x value is performed when this counter is decrementing.

Note: This bit is not relevant if the counter of the channel 0 is left-aligned (CALG = 0 in PWM Channel Mode Register)

CAUTION: The write of the register PWM_CMPVUPDx must be followed by a write of the register PWM_CMPMUPDx.

Downloaded from Arrow.com.

49.7.38 PWM Comparison x Mode Register

Name: PWM_CMPMx

Address: 0x40020138 (0)[0], 0x40020148 (0)[1], 0x40020158 (0)[2], 0x40020168 (0)[3], 0x40020178 (0)[4], 0x40020188 (0)[5], 0x40020198 (0)[6], 0x400201A8 (0)[7], 0x4005C138 (1)[0], 0x4005C148 (1)[1], 0x4005C158 (1)[2], 0x4005C168 (1)[3], 0x4005C178 (1)[4], 0x4005C188 (1)[5], 0x4005C198 (1)[6], 0x4005C1A8 (1)[7]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
CUPRCNT				CUPR			

15	14	13	12	11	10	9	8
CPRCNT				CPR			

7	6	5	4	3	2	1	0
CTR				–	–	–	CEN

- **CEN: Comparison x Enable**

0: The comparison x is disabled and can not match.

1: The comparison x is enabled and can match.

- **CTR: Comparison x Trigger**

The comparison x is performed when the value of the comparison x period counter (CPRCNT) reaches the value defined by CTR.

- **CPR: Comparison x Period**

CPR defines the maximum value of the comparison x period counter (CPRCNT). The comparison x value is performed periodically once every CPR+1 periods of the channel 0 counter.

- **CPRCNT: Comparison x Period Counter**

Reports the value of the comparison x period counter.

Note: The field CPRCNT is read-only

- **CUPR: Comparison x Update Period**

Defines the time between each update of the comparison x mode and the comparison x value. This time is equal to CUPR+1 periods of the channel 0 counter.

- **CUPRCNT: Comparison x Update Period Counter**

Reports the value of the comparison x update period counter.

Note: The field CUPRCNT is read-only

Downloaded from Arrow.com.

49.7.39 PWM Comparison x Mode Update Register

Name: PWM_CMPMUPDx

Address: 0x4002013C (0)[0], 0x4002014C (0)[1], 0x4002015C (0)[2], 0x4002016C (0)[3], 0x4002017C (0)[4], 0x4002018C (0)[5], 0x4002019C (0)[6], 0x400201AC (0)[7], 0x4005C13C (1)[0], 0x4005C14C (1)[1], 0x4005C15C (1)[2], 0x4005C16C (1)[3], 0x4005C17C (1)[4], 0x4005C18C (1)[5], 0x4005C19C (1)[6], 0x4005C1AC (1)[7]

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	CUPRUPD			

15	14	13	12	11	10	9	8
–	–	–	–	CPRUPD			

7	6	5	4	3	2	1	0
CTRUPD				–	–	–	CENUPD

This register acts as a double buffer for the CEN, CTR, CPR and CUPR values. This prevents an unexpected comparison x match.

- **CENUPD: Comparison x Enable Update**

0: The comparison x is disabled and can not match.

1: The comparison x is enabled and can match.

- **CTRUPD: Comparison x Trigger Update**

The comparison x is performed when the value of the comparison x period counter (CPRCNT) reaches the value defined by CTR.

- **CPRUPD: Comparison x Period Update**

CPR defines the maximum value of the comparison x period counter (CPRCNT). The comparison x value is performed periodically once every CPR+1 periods of the channel 0 counter.

- **CUPRUPD: Comparison x Update Period Update**

Defines the time between each update of the comparison x mode and the comparison x value. This time is equal to CUPR+1 periods of the channel 0 counter.

Downloaded from Arrow.com.

49.7.40 PWM Channel Mode Register

Name: PWM_CMRx [x=0..3]

Address: 0x40020200 (0)[0], 0x40020220 (0)[1], 0x40020240 (0)[2], 0x40020260 (0)[3], 0x4005C200 (1)[0], 0x4005C220 (1)[1], 0x4005C240 (1)[2], 0x4005C260 (1)[3]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	PPM	DTLI	DTHI	DTE
15	14	13	12	11	10	9	8
–	–	TCTS	DPOLI	UPDS	CES	CPOL	CALG
7	6	5	4	3	2	1	0
–	–	–	–	CPRE			

This register can only be written if bits WPSWS2 and WPHWS2 are cleared in the PWM Write Protection Status Register.

- **CPRE: Channel Pre-scaler**

Value	Name	Description
0	MCK	Peripheral clock
1	MCK_DIV_2	Peripheral clock/2
2	MCK_DIV_4	Peripheral clock/4
3	MCK_DIV_8	Peripheral clock/8
4	MCK_DIV_16	Peripheral clock/16
5	MCK_DIV_32	Peripheral clock/32
6	MCK_DIV_64	Peripheral clock/64
7	MCK_DIV_128	Peripheral clock/128
8	MCK_DIV_256	Peripheral clock/256
9	MCK_DIV_512	Peripheral clock/512
10	MCK_DIV_1024	Peripheral clock/1024
11	CLKA	Clock A
12	CLKB	Clock B

- **CALG: Channel Alignment**

0: The period is left-aligned.

1: The period is center-aligned.

- **CPOL: Channel Polarity**

0: The OCx output waveform (output from the comparator) starts at a low level.

1: The OCx output waveform (output from the comparator) starts at a high level.

Downloaded from Arrow.com.

- **CES: Counter Event Selection**

The bit CES defines when the channel counter event occurs when the period is center-aligned (flag CHIDx in PWM Interrupt Status Register 1).

CALG = 0 (Left Alignment):

0/1: The channel counter event occurs at the end of the PWM period.

CALG = 1 (Center Alignment):

0: The channel counter event occurs at the end of the PWM period.

1: The channel counter event occurs at the end of the PWM period and at half the PWM period.

- **UPDS: Update Selection**

When the period is center aligned, the bit UPDS defines when the update of the duty cycle, the polarity value/mode occurs after writing the corresponding update registers.

CALG = 0 (Left Alignment):

0/1: The update always occurs at the end of the PWM period after writing the update register(s).

CALG = 1 (Center Alignment):

0: The update occurs at the next end of the PWM period after writing the update register(s).

1: The update occurs at the next end of the PWM half period after writing the update register(s).

- **DPOLI: Disabled Polarity Inverted**

0: When the PWM channel x is disabled (CHIDx(PWM_SR) = 0), the OCx output waveform is the same as the one defined by the CPOL bit.

1: When the PWM channel x is disabled (CHIDx(PWM_SR) = 0), the OCx output waveform is inverted compared to the one defined by the CPOL bit.

- **TCTS: Timer Counter Trigger Selection**

0: The comparator of the channel x (OCx) is used as the trigger source for the Timer Counter (TC).

1: The counter events of the channel x is used as the trigger source for the Timer Counter (TC).

- **DTE: Dead-Time Generator Enable**

0: The dead-time generator is disabled.

1: The dead-time generator is enabled.

- **DTHI: Dead-Time PWMHx Output Inverted**

0: The dead-time PWMHx output is not inverted.

1: The dead-time PWMHx output is inverted.

- **DTLI: Dead-Time PWMLx Output Inverted**

0: The dead-time PWMLx output is not inverted.

1: The dead-time PWMLx output is inverted.

- **PPM: Push-Pull Mode**

0: The Push-Pull mode is disabled for channel x.

1: The Push-Pull mode is enabled for channel x.

Downloaded from Arrow.com.

49.7.41 PWM Channel Duty Cycle Register

Name: PWM_CDTYx [x=0..3]

Address: 0x40020204 (0)[0], 0x40020224 (0)[1], 0x40020244 (0)[2], 0x40020264 (0)[3], 0x4005C204 (1)[0], 0x4005C224 (1)[1], 0x4005C244 (1)[2], 0x4005C264 (1)[3]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
CDTY							

15	14	13	12	11	10	9	8
CDTY							

7	6	5	4	3	2	1	0
CDTY							

Only the first 16 bits (channel counter size) are significant.

- **CDTY: Channel Duty-Cycle**

Defines the waveform duty-cycle. This value must be defined between 0 and CPRD (PWM_CPRDx).

Downloaded from Arrow.com.

49.7.42 PWM Channel Duty Cycle Update Register

Name: PWM_CDTYUPDx [x=0..3]

Address: 0x40020208 (0)[0], 0x40020228 (0)[1], 0x40020248 (0)[2], 0x40020268 (0)[3], 0x4005C208 (1)[0], 0x4005C228 (1)[1], 0x4005C248 (1)[2], 0x4005C268 (1)[3]

Access: Write-only.

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
CDTYUPD							

15	14	13	12	11	10	9	8
CDTYUPD							

7	6	5	4	3	2	1	0
CDTYUPD							

This register acts as a double buffer for the CDTY value. This prevents an unexpected waveform when modifying the waveform duty-cycle.

Only the first 16 bits (channel counter size) are significant.

- **CDTYUPD: Channel Duty-Cycle Update**

Defines the waveform duty-cycle. This value must be defined between 0 and CPRD (PWM_CPRDx).

Downloaded from Arrow.com.

49.7.43 PWM Channel Period Register

Name: PWM_CPRDx [x=0..3]

Address: 0x4002020C (0)[0], 0x4002022C (0)[1], 0x4002024C (0)[2], 0x4002026C (0)[3], 0x4005C20C (1)[0], 0x4005C22C (1)[1], 0x4005C24C (1)[2], 0x4005C26C (1)[3]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
CPRD							

15	14	13	12	11	10	9	8
CPRD							

7	6	5	4	3	2	1	0
CPRD							

This register can only be written if bits WPSWS3 and WPHWS3 are cleared in the PWM Write Protection Status Register.

Only the first 16 bits (channel counter size) are significant.

- **CPRD: Channel Period**

If the waveform is left-aligned, then the output waveform period depends on the channel counter source clock and can be calculated:

– By using the PWM peripheral clock divided by a given prescaler value “X” (where X = 2^{PREA} is 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula is:

$$\frac{(X \times CPRD)}{f_{peripheral\ clock}}$$

– By using the PWM peripheral clock divided by a given prescaler value “X” (see above) and by either the DIVA or the DIVB divider. The formula becomes, respectively:

$$\frac{(X \times CPRD \times DIVA)}{f_{peripheral\ clock}} \ or \ \frac{(X \times CPRD \times DIVB)}{f_{peripheral\ clock}}$$

If the waveform is center-aligned, then the output waveform period depends on the channel counter source clock and can be calculated:

– By using the PWM peripheral clock divided by a given prescaler value “X” (where X = 2^{PREA} is 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula is:

$$\frac{(2 \times X \times CPRD)}{f_{peripheral\ clock}}$$

– By using the PWM peripheral clock divided by a given prescaler value “X” (see above) and by either the DIVA or the DIVB divider. The formula becomes, respectively:

$$\frac{(2 \times X \times CPRD \times DIVA)}{f_{peripheral\ clock}} \ or \ \frac{(2 \times X \times CPRD \times DIVB)}{f_{peripheral\ clock}}$$

Downloaded from Arrow.com.

49.7.44 PWM Channel Period Update Register

Name: PWM_CPRDUPDx [x=0..3]

Address: 0x40020210 (0)[0], 0x40020230 (0)[1], 0x40020250 (0)[2], 0x40020270 (0)[3], 0x4005C210 (1)[0], 0x4005C230 (1)[1], 0x4005C250 (1)[2], 0x4005C270 (1)[3]

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
CPRDUPD							
15	14	13	12	11	10	9	8
CPRDUPD							
7	6	5	4	3	2	1	0
CPRDUPD							

This register can only be written if bits WPSWS3 and WPHWS3 are cleared in the PWM Write Protection Status Register.

This register acts as a double buffer for the CPRD value. This prevents an unexpected waveform when modifying the waveform period.

Only the first 16 bits (channel counter size) are significant.

- **CPRDUPD: Channel Period Update**

If the waveform is left-aligned, then the output waveform period depends on the channel counter source clock and can be calculated:

– By using the PWM peripheral clock divided by a given prescaler value “X” (where X = 2^{PREA} is 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula is:

$$\frac{(X \times CPRDUPD)}{f_{peripheral\ clock}}$$

– By using the PWM peripheral clock divided by a given prescaler value “X” (see above) and by either the DIVA or the DIVB divider. The formula becomes, respectively:

$$\frac{(X \times CRPDUPD \times DIVA)}{f_{peripheral\ clock}} \ or \ \frac{(X \times CRPDUPD \times DIVB)}{f_{peripheral\ clock}}$$

If the waveform is center-aligned, then the output waveform period depends on the channel counter source clock and can be calculated:

– By using the PWM peripheral clock divided by a given prescaler value “X” (where X = 2^{PREA} is 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula is:

$$\frac{(2 \times X \times CPRDUPD)}{f_{peripheral\ clock}}$$

– By using the PWM peripheral clock divided by a given prescaler value “X” (see above) and by either the DIVA or the DIVB divider. The formula becomes, respectively:

$$\frac{(2 \times X \times CPRDUPD \times DIVA)}{f_{peripheral\ clock}} \text{ or } \frac{(2 \times X \times CPRDUPD \times DIVB)}{f_{peripheral\ clock}}$$

Downloaded from Arrow.com.

49.7.45 PWM Channel Counter Register

Name: PWM_CCNTx [x=0..3]

Address: 0x40020214 (0)[0], 0x40020234 (0)[1], 0x40020254 (0)[2], 0x40020274 (0)[3], 0x4005C214 (1)[0], 0x4005C234 (1)[1], 0x4005C254 (1)[2], 0x4005C274 (1)[3]

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
CNT							

15	14	13	12	11	10	9	8
CNT							

7	6	5	4	3	2	1	0
CNT							

Only the first 16 bits (channel counter size) are significant.

- **CNT: Channel Counter Register**

Channel counter value. This register is reset when:

- the channel is enabled (writing CHIDx in the PWM_ENA register).
- the channel counter reaches CPRD value defined in the PWM_CPRDx register if the waveform is left-aligned.

Downloaded from Arrow.com.

49.7.46 PWM Channel Dead Time Register

Name: PWM_DTx [x=0..3]

Address: 0x40020218 (0)[0], 0x40020238 (0)[1], 0x40020258 (0)[2], 0x40020278 (0)[3], 0x4005C218 (1)[0], 0x4005C238 (1)[1], 0x4005C258 (1)[2], 0x4005C278 (1)[3]

Access: Read/Write

31	30	29	28	27	26	25	24
DTL							

23	22	21	20	19	18	17	16
DTL							

15	14	13	12	11	10	9	8
DTH							

7	6	5	4	3	2	1	0
DTH							

This register can only be written if bits WPSWS4 and WPHWS4 are cleared in the PWM Write Protection Status Register.

Only the first 12 bits (dead-time counter size) of fields DTH and DTL are significant.

- **DTH: Dead-Time Value for PWMHx Output**

Defines the dead-time value for PWMHx output. This value must be defined between 0 and the value (CPRD – CDTY) (PWM_CPRDx and PWM_CDTYx).

- **DTL: Dead-Time Value for PWMLx Output**

Defines the dead-time value for PWMLx output. This value must be defined between 0 and CDTY (PWM_CDTYx).

Downloaded from Arrow.com.

49.7.47 PWM Channel Dead Time Update Register

Name: PWM_DTUPDx [x=0..3]

Address: 0x4002021C (0)[0], 0x4002023C (0)[1], 0x4002025C (0)[2], 0x4002027C (0)[3], 0x4005C21C (1)[0], 0x4005C23C (1)[1], 0x4005C25C (1)[2], 0x4005C27C (1)[3]

Access: Write-only

31	30	29	28	27	26	25	24
DTLUPD							

23	22	21	20	19	18	17	16
DTLUPD							

15	14	13	12	11	10	9	8
DTHUPD							

7	6	5	4	3	2	1	0
DTHUPD							

This register can only be written if bits WPSWS4 and WPHWS4 are cleared in the PWM Write Protection Status Register.

This register acts as a double buffer for the DTH and DTL values. This prevents an unexpected waveform when modifying the dead-time values.

Only the first 12 bits (dead-time counter size) of fields DTHUPD and DTLUPD are significant.

- **DTHUPD: Dead-Time Value Update for PWMHx Output**

Defines the dead-time value for PWMHx output. This value must be defined between 0 and the value (CPRD – CDTY) (PWM_CPRDx and PWM_CDTYx). This value is applied only at the beginning of the next channel x PWM period.

- **DTLUPD: Dead-Time Value Update for PWMLx Output**

Defines the dead-time value for PWMLx output. This value must be defined between 0 and CDTY (PWM_CDTYx). This value is applied only at the beginning of the next channel x PWM period.

Downloaded from Arrow.com.

49.7.48 PWM Channel Mode Update Register

Name: PWM_CMUPDx [x=0..3]

Address: 0x40020400 (0)[0], 0x40020420 (0)[1], 0x40020440 (0)[2], 0x40020460 (0)[3], 0x4005C400 (1)[0], 0x4005C420 (1)[1], 0x4005C440 (1)[2], 0x4005C460 (1)[3]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	CPOLINVUP	–	–	–	CPOLUP	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

This register can only be written if bits WPSWS2 and WPHWS2 are cleared in the PWM Write Protection Status Register.

This register acts as a double buffer for the CPOL value. This prevents an unexpected waveform when modifying the polarity value.

- **CPOLUP: Channel Polarity Update**

The write of this bit is taken into account only if the bit CPOLINVUP is written at ‘0’ at the same time.

0: The OCx output waveform (output from the comparator) starts at a low level.

1: The OCx output waveform (output from the comparator) starts at a high level.

- **CPOLINVUP: Channel Polarity Inversion Update**

If this bit is written at ‘1’, the write of the bit CPOLUP is not taken into account.

0: No effect.

1: The OCx output waveform (output from the comparator) is inverted.

Downloaded from Arrow.com.

49.7.49 PWM External Trigger Register

Name: PWM_ETRGx [x=1..2]

Address: 0x4002042C (0)[1], 0x4002044C (0)[2], 0x4005C42C (1)[1], 0x4005C44C (1)[2]

Access: Read/Write

31	30	29	28	27	26	25	24
RFEN	TRGSRC	TRGFILT	TRGEDGE	–	–	TRGMODE	

23	22	21	20	19	18	17	16
MAXCNT							

15	14	13	12	11	10	9	8
MAXCNT							

7	6	5	4	3	2	1	0
MAXCNT							

- **MAXCNT: Maximum Counter value**

Maximum channel x counter value measured at the TRGINx event since the last read of the register.

At the TRGINx event, if the channel x counter value is greater than the stored MAXCNT value, then MAXCNT is updated by the channel x counter value.

- **TRGMODE: External Trigger Mode**

Value	Name	Description
0	OFF	External trigger is not enabled.
1	MODE1	External PWM Reset Mode
2	MODE2	External PWM Start Mode
3	MODE3	Cycle-by-cycle Duty Mode

- **TRGEDGE: Edge Selection**

Value	Name	Description
0	FALLING_ZERO	TRGMODE = 1: TRGINx event detection on falling edge. TRGMODE = 2, 3: TRGINx active level is 0
1	RISING_ONE	TRGMODE = 1: TRGINx event detection on rising edge. TRGMODE = 2, 3: TRGINx active level is 1

- **TRGFILT: Filtered input**

0: The external trigger input x is not filtered.

1: The external trigger input x is filtered.

- **RFEN: Recoverable Fault Enable**

0: The TRGINx signal does not generate a recoverable fault.

1: The TRGINx signal generate a recoverable fault in place of the fault x input.

Downloaded from Arrow.com.

- **TRGSRC: Trigger Source**

0: The TRGINx signal is driven by the PWMTRGx input.

1: The TRGINx signal is driven by the Analog Comparator Controller.

Downloaded from Arrow.com.

49.7.50 PWM Leading-Edge Blanking Register

Name: PWM_LEBRx [x=1..2]

Address: 0x40020430 (0)[1], 0x40020450 (0)[2], 0x4005C430 (1)[1], 0x4005C450 (1)[2]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	PWMHREN	PWMHFEN	PWMLREN	PWMLFEN
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	LEBDELAY						

- **LEBDELAY: Leading-Edge Blanking Delay for TRGINx**

Leading-edge blanking duration for external trigger x input. The delay is calculated according to the following formula:

$$LEBDELAY = (f_{peripheral\ clock} \times Delay) + 1$$

- **PWMLFEN: PWML Falling Edge Enable**

0: Leading-edge blanking is disabled on PWMLx output falling edge.

1: Leading-edge blanking is enabled on PWMLx output falling edge.

- **PWMLREN: PWML Rising Edge Enable**

0: Leading-edge blanking is disabled on PWMLx output rising edge.

1: Leading-edge blanking is enabled on PWMLx output rising edge.

- **PWMHFEN: PWMH Falling Edge Enable**

0: Leading-edge blanking is disabled on PWMHx output falling edge.

1: Leading-edge blanking is enabled on PWMHx output falling edge.

- **PWMHREN: PWMH Rising Edge Enable**

0: Leading-edge blanking is disabled on PWMHx output rising edge.

1: Leading-edge blanking is enabled on PWMHx output rising edge.

Downloaded from Arrow.com.

50. Analog Front-End Controller (AFEC)

50.1 Description

The Analog Front-End Controller (AFEC) is based on an Analog Front-End cell (AFE) integrating a 12-bit Analog-to-Digital Converter (ADC), a Programmable Gain Amplifier (PGA), a Digital-to-Analog Converter (DAC) and two 6-to-1 analog multiplexers, making possible the analog-to-digital conversions of 12 analog lines (in single Sample-and-Hold mode) or two simultaneous conversions of 6 analog lines (in dual Sample-and-Hold mode). The conversions extend from 0V to VREFP. The AFEC supports an 12-bit resolution mode which can be extended up to a 16-bit resolution by digital averaging.

Conversion results are reported in a common register for all channels, as well as in a channel-dedicated register.

Software trigger, external trigger on rising edge of the AFE_ADTRG pin or internal triggers from Timer Counter output(s) are configurable.

The comparison circuitry allows automatic detection of values below a threshold, higher than a threshold, in a given range or outside the range. Thresholds and ranges are fully configurable.

The AFEC internal fault output is directly connected to PWM Fault input. This input can be asserted by means of comparison circuitry in order to immediately put the PWM outputs in a safe state (pure combinational path).

The AFEC also integrates a Sleep mode and a conversion sequencer and connects with a DMA channel. These features reduce both power consumption and processor intervention.

The AFEC has a selectable single-ended or fully differential input and benefits from a 2-bit programmable gain. A set of reference voltages is generated internally from a single external reference voltage node that may be equal to the analog supply voltage. An external decoupling capacitance is required for noise filtering.

A digital error correction circuit based on the multi-bit redundant signed digit (RSD) algorithm is employed in order to reduce INL and DNL errors.

Finally, the user can configure AFE timings, such as startup time and tracking time.

Downloaded from Arrow.com.

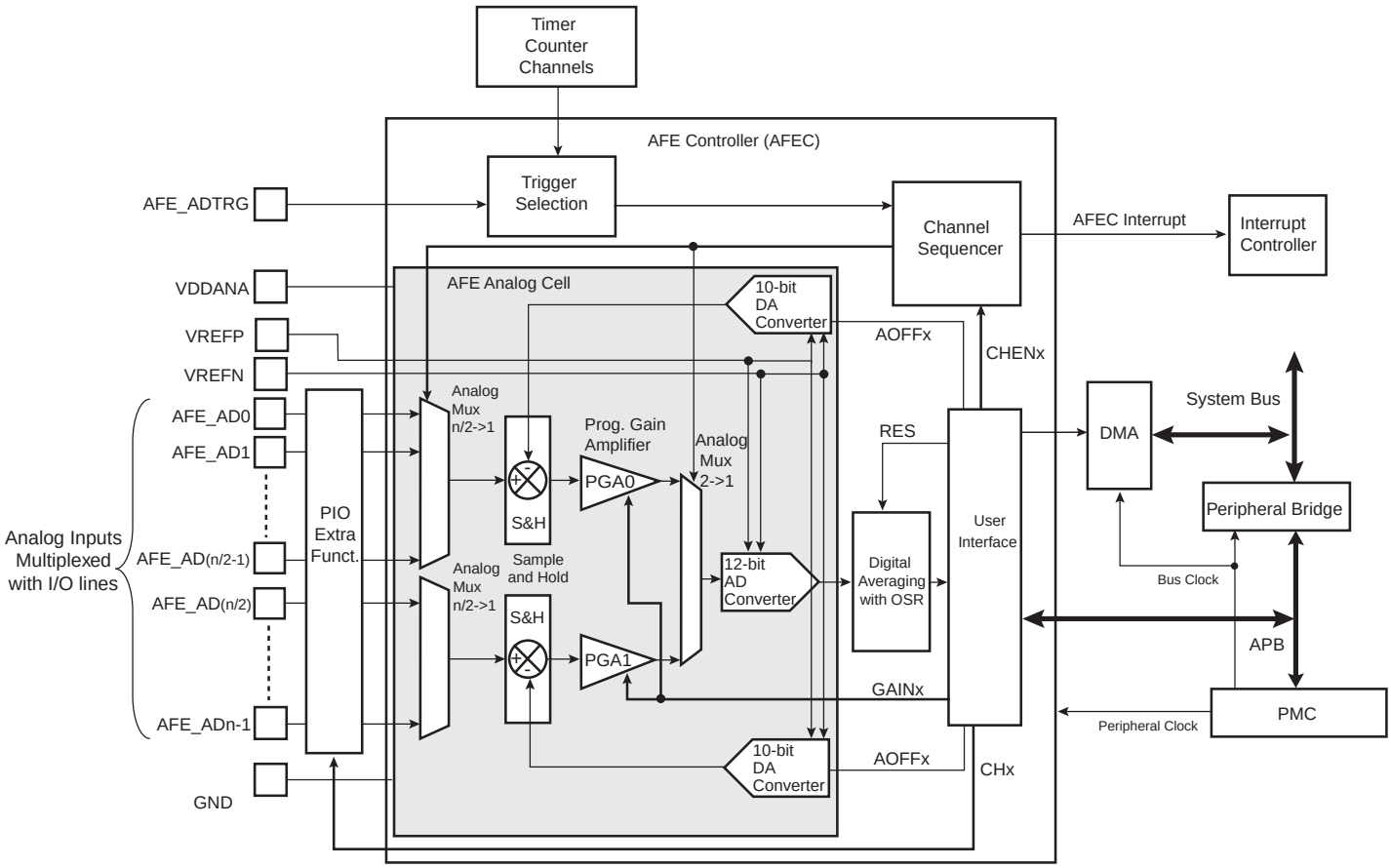
50.2 Embedded Characteristics

- 12-bit resolution up to 16-bit resolution by digital averaging
- 2MHz Conversion Rate
- Wide Range of Power Supply Operation
- Selectable Single-ended or Differential Input Voltage
- Selectable Single or Dual Sample-and-Hold Mode
- Programmable Gain for Maximum Full-Scale Input Range 0–V_{DD}
- Programmable Offset Per Channel
- Automatic correction of offset and gain errors
- Integrated Multiplexers Offering Up to 12 Independent Analog Inputs
- Individual Enable and Disable of Each Channel
- Hardware or Software Trigger
  - External trigger pin
  - Timer counter outputs (corresponding TIOA trigger)
  - PWM event line
- Drive of PWM Fault Input
- DMA Support
- Possibility of AFE Timings Configuration
- Two Sleep Modes and Conversion Sequencer
  - Automatic wake-up on trigger and back to sleep mode after conversions of all enabled channels
  - Possibility of customized channel sequence
- Standby Mode for Fast Wake-up Time Response
  - Power-down capability
- Automatic Window Comparison of Converted Values
- Register Write Protection

Downloaded from Arrow.com.

50.3 Block Diagram

Figure 50-1. Analog Front-End Controller Block Diagram



50.4 Signal Description

Table 50-1. AFEC Signal Description

Pin Name	Description
VREFP	Reference voltage
VREFN	Reference voltage
AFE_AD0—AFE_AD11[1]	Analog input channels
AFE_ADTRG	External trigger

Note: 1. AFE_AD11 is not an actual pin but is connected to a temperature sensor.

Downloaded from Arrow.com.

50.5 Product Dependencies

50.5.1 I/O Lines

The digital input AFE_ADTRG is multiplexed with digital functions on the I/O line and the selection of AFE_ADTRG is made using the PIO Controller.

The analog inputs AFE_ADx are multiplexed with digital functions on the I/O lines. AFE_ADx inputs are selected as inputs of the AFEC when writing a one in the corresponding CHx bit of AFEC_CHER and the digital functions are not selected.

Table 50-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
AFEC0	AFE0_ADTRG	PA8	B
AFEC0	AFE0_AD0	PD30	X1
AFEC0	AFE0_AD1/PIODCEN2	PA21	X1
AFEC0	AFE0_AD2/WKUP12	PB3	X1
AFEC0	AFE0_AD3	PE5	X1
AFEC0	AFE0_AD4	PE4	X1
AFEC0	AFE0_AD5	PB2	X1
AFEC0	AFE0_AD6	PA17	X1
AFEC0	AFE0_AD7	PA18	X1
AFEC0	AFE0_AD8/WKUP9	PA19	X1
AFEC0	AFE0_AD9/WKUP10	PA20	X1
AFEC0	AFE0_AD10/RTCOUT0	PB0	X1
AFEC1	AFE1_ADTRG	PD9	C
AFEC1	AFE1_AD0/RTCOUT1	PB1	X1
AFEC1	AFE1_AD1	PC13	X1
AFEC1	AFE1_AD2	PC15	X1
AFEC1	AFE1_AD3	PC12	X1
AFEC1	AFE1_AD4	PC29	X1
AFEC1	AFE1_AD5	PC30	X1
AFEC1	AFE1_AD6	PC31	X1
AFEC1	AFE1_AD7	PC26	X1
AFEC1	AFE1_AD8	PC27	X1
AFEC1	AFE1_AD9	PC0	X1
AFEC1	AFE1_AD10	PE3	X1
AFEC1	AFE1_AD11	PE0	X1

50.5.2 Power Management

The AFEC is not continuously clocked. The programmer must first enable the AFEC peripheral clock in the Power Management Controller (PMC) before using the AFEC. However, if the application does not require AFEC operations, the peripheral clock can be stopped when not needed and restarted when necessary.

When the AFEC is in Sleep mode, the peripheral clock must always be enabled.

Downloaded from Arrow.com.

50.5.3 Interrupt Sources

The AFEC interrupt line is connected on one of the internal sources of the Interrupt Controller. Using the AFEC interrupt requires the interrupt controller to be programmed first.

Table 50-3. Peripheral IDs

Instance	ID
AFEC0	29
AFEC1	40

50.5.4 Temperature Sensor

The temperature sensor is connected to Channel 11 of the AFEC.

The temperature sensor provides an output voltage V_T that is proportional to the absolute temperature (PTAT).

50.5.5 Timer Triggers

Timer Counters may or may not be used as hardware triggers depending on user requirements. Thus, some or all of the timer counters may be unconnected.

50.5.6 PWM Event Line

PWM event lines may or may not be used as hardware triggers depending on user requirements.

50.5.7 Fault Output

The AFEC has the Fault output connected to the FAULT input of PWM. Refer to Section 50.6.17 ”Fault Output” and implementation of the PWM in the product.

50.5.8 Conversion Performances

For performance and electrical characteristics of the AFE, refer to the AFE Characteristics in Section 56. ”Electrical Characteristics”.

Downloaded from Arrow.com.

50.6 Functional Description

50.6.1 Analog Front-End Conversion

The AFE embeds programmable gain amplifiers that must be enabled prior to any conversion. The bits PGA0EN and PGA1EN in the Analog Control register (AFEC_ACR) must be set.

The AFE uses the AFE clock to perform conversions. In order to guarantee a conversion with minimum error, after any start of conversion, the AFEC waits a number of AFE clock cycles (called transfer time) before changing the channel selection again (and so starts a new tracking operation).

AFE conversions are sequenced by two operating times: the tracking time and the conversion time.

- The tracking time represents the time between the channel selection change and the time for the controller to start the AFEC. The AFEC allows a minimum tracking time of 15 AFE clock periods.
- The conversion time represents the time for the AFEC to convert the analog signal.

The AFE clock frequency is selected in the PRESCAL field of the AFEC_MR. The tracking phase starts during the conversion of the previous channel. If the tracking time is longer than the conversion time of the12-bit AD converter (t_{CONV}), the tracking phase is extended to the end of the previous conversion.

The AFE clock frequency ranges from $f_{peripheral\ clock}/2$ if PRESCAL is 1, and $f_{peripheral\ clock}/512$ if PRESCAL is set to 255 (0xFF). PRESCAL must be programmed to provide the AFE clock frequency given in the section “Electrical Characteristics”.

The AFE conversion time (t_{AFE_conv}) is applicable for all modes and is calculated as follows:

$$t_{AFE_conv} = 21 \times t_{AFE\ Clock}$$

When the averager is activated, the AFE conversion time is multiplied by the OSR value.

In Free Run mode, the sampling frequency (f_S) is calculated as $1/t_{AFE_conv}$.

Downloaded from Arrow.com.

Figure 50-2. Sequence of AFE Conversions when Tracking Time > Conversion Time

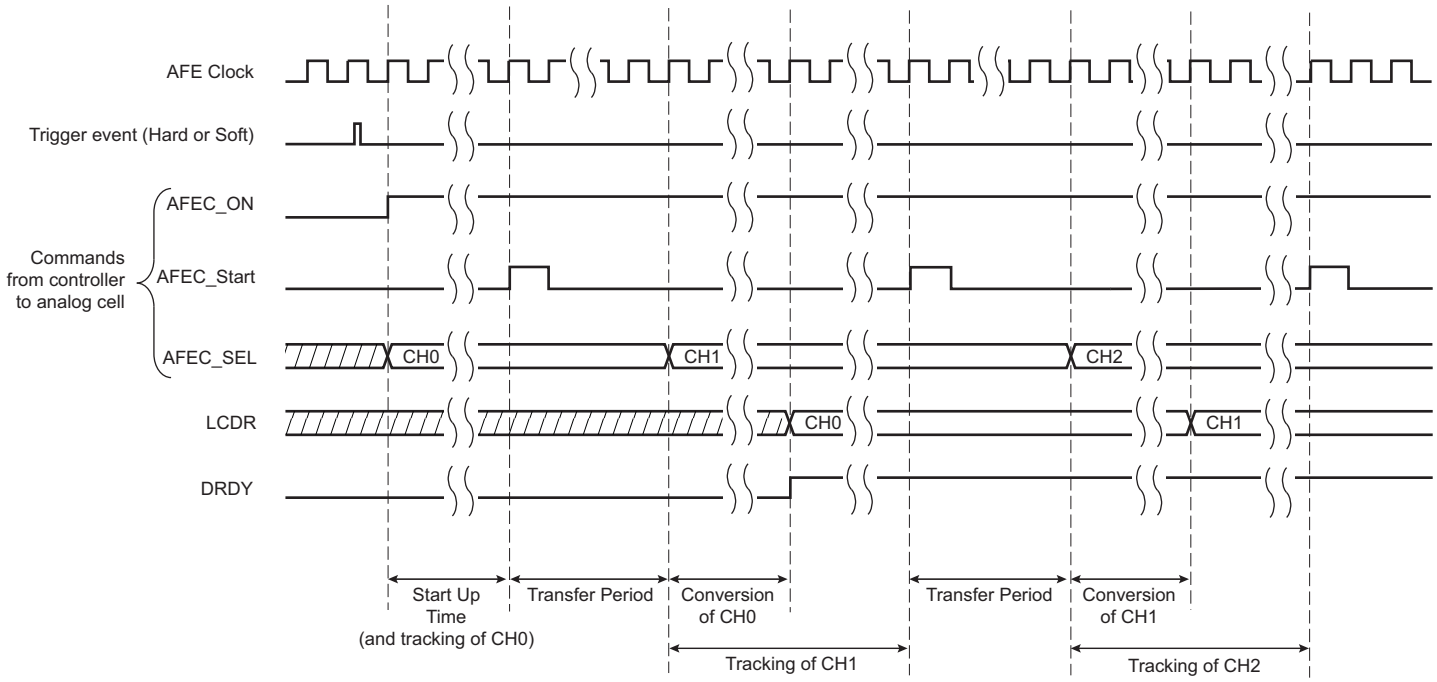
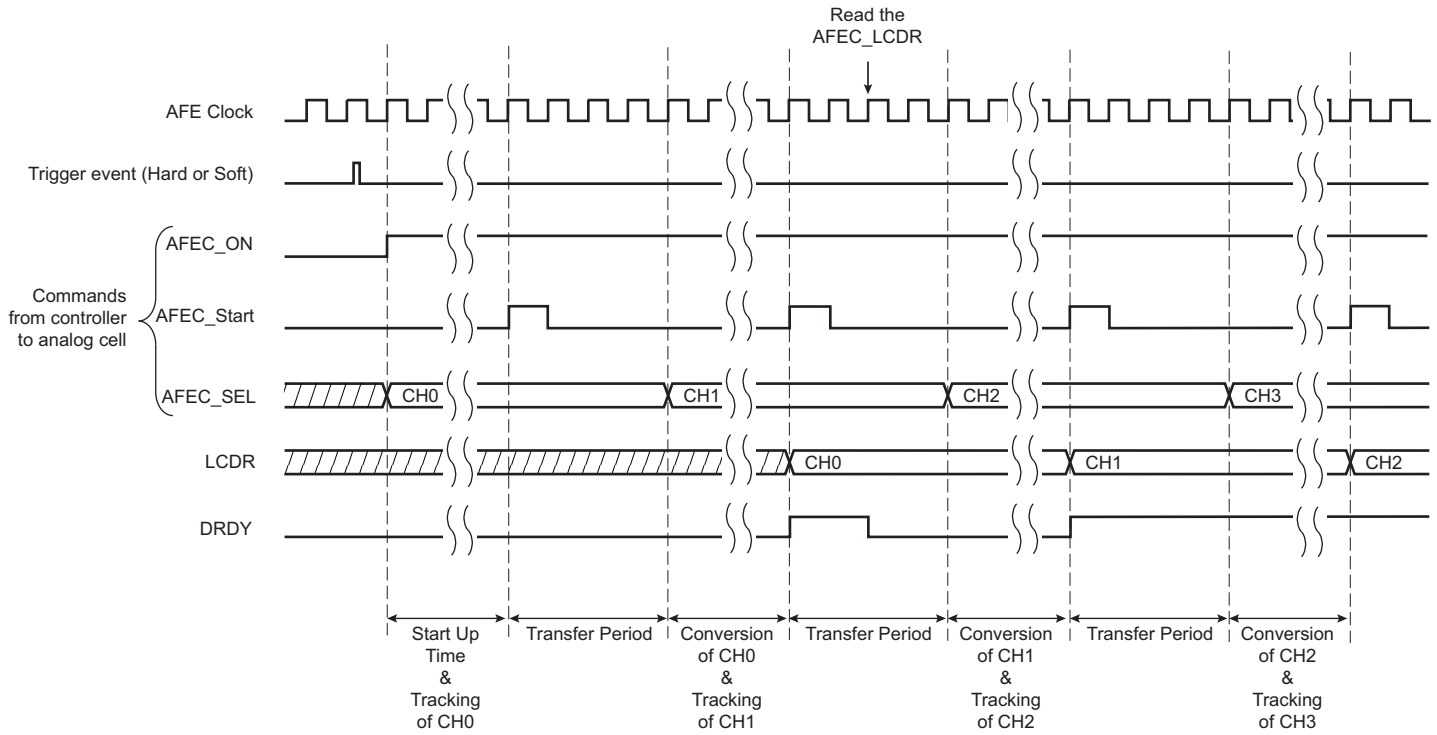


Figure 50-3. Sequence of AFE Conversions when Tracking Time < Conversion Time



Downloaded from Arrow.com.

50.6.2 Conversion Reference

The conversion is performed on a full range between 0V and the reference voltage carried on pin VREFP. Analog inputs between these voltages convert to values based on a linear conversion.

50.6.3 Conversion Resolution

The AFEC supports 12-bit native resolutions. Writing two or more to the RES field in the Extended Mode register (AFEC_EMR) automatically enables the Enhanced Resolution mode. For details on this mode, see Section 50.6.14.

Moreover, when a DMA channel is connected to the AFEC, a resolution lower than 16 bits sets the transfer request size to 16 bits.

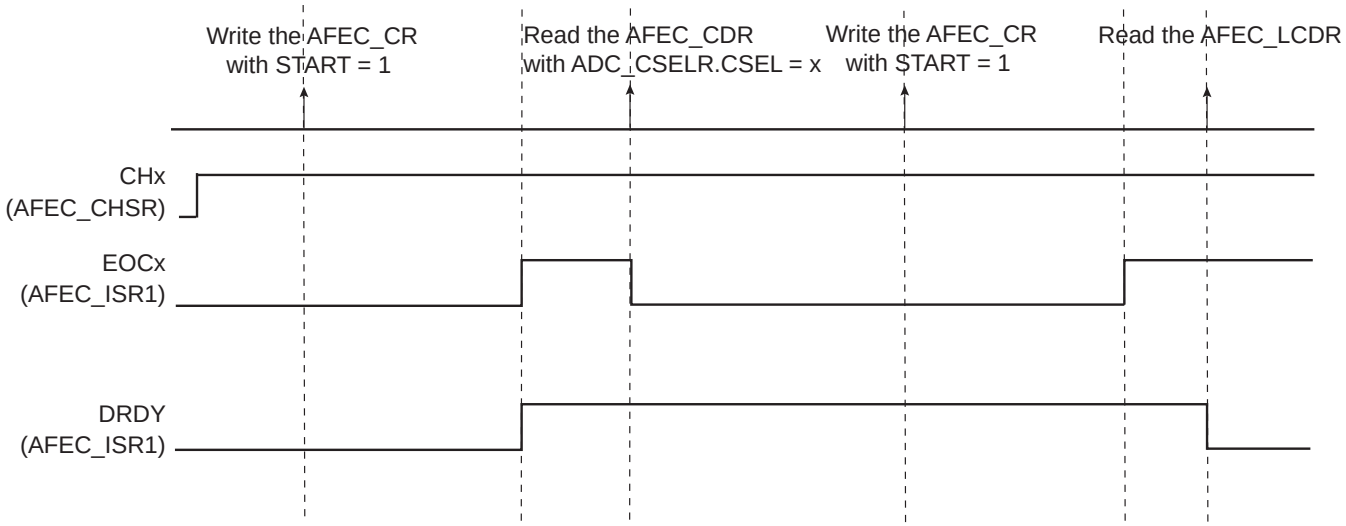
50.6.4 Conversion Results

When a conversion is completed, the resulting 12-bit digital value is stored in an internal register (one register for each channel) that can be read by means of the Channel Data Register (AFEC_CDR) and the Last Converted Data Register (AFEC_LCDR). By setting the bit TAG in the AFEC_EMR, the AFEC_LCDR presents the channel number associated with the last converted data in the CHNB field.

The bits EOCx, where ‘x’ corresponds to the value programmed in the CSEL bit of AFEC_CSELR, and DRDY in the Interrupt Status Register (AFEC_ISR) are set. In the case of a connected DMA channel, DRDY rising triggers a data transfer request. In any case, either EOCx or DRDY can trigger an interrupt.

Reading the AFEC_CDR clears the EOCx bit. Reading AFEC_LCDR clears the DRDY bit and the EOCx bit corresponding to the last converted channel.

Figure 50-4. EOCx and DRDY Flag Behavior



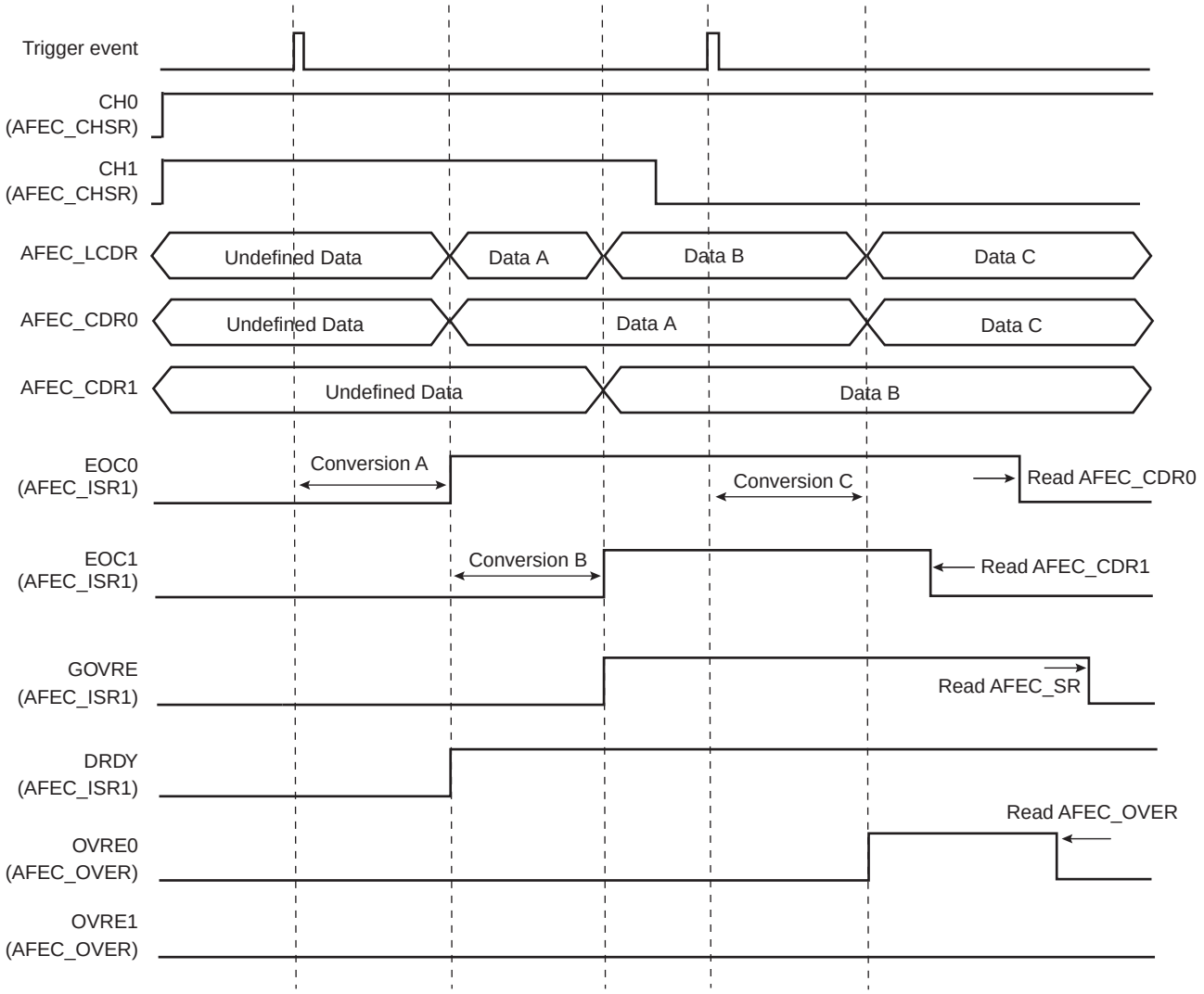
If AFEC_CDR is not read before further incoming data is converted, the corresponding OVREx flag is set in the Overrun Status Register (AFEC_OVER).

New data converted when DRDY is high sets the GOVRE bit in AFEC_ISR.

The OVREx flag is automatically cleared when AFEC_OVER is read, and the GOVRE flag is automatically cleared when AFEC_ISR is read.

Downloaded from Arrow.com.

Figure 50-5. EOCx, GOVRE and OVREx Flag Behavior



Warning: If the corresponding channel is disabled during a conversion, or if it is disabled and then reenabled during a conversion, its associated data and its corresponding EOCx and GOVRE flags in AFEC_ISR and OVREx flags in AFEC_OVER are unpredictable.

Downloaded from Arrow.com.

50.6.5 Conversion Results Format

The conversion results can be signed (2's complement) or unsigned depending on the value of the SIGNMODE field in AFEC_EMR.

Four modes are available:

- Results of channels configured in single-ended mode are unsigned; results of channels configured in differential mode are signed.
- Results of channels configured in single-ended mode are signed; results of channels configured in differential mode are unsigned.
- Results of all channels are unsigned.
- Results of all channels are signed.

If conversion results are signed and resolution is less than 16 bits, the sign is extended up to the bit 15 (e.g., 0xF43 for 12-bit resolution will be read as 0xFF43 and 0x467 will be read as 0x0467).

50.6.6 Conversion Triggers

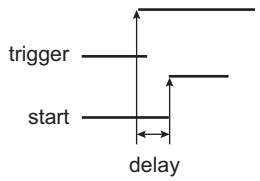
Conversions of the active analog channels are started with a software or hardware trigger. The software trigger is provided by writing the Control Register (AFEC_CR) with the START bit at 1.

The hardware trigger can be one of the TIOA outputs of the Timer Counter channels, PWM Event line, or the external trigger input of the AFEC (ADTRG). The hardware trigger is selected with the TRGSEL field in the AFEC_MR. The selected hardware trigger is enabled with the TRGEN bit in the AFEC_MR.

The minimum time between two consecutive trigger events must be strictly greater than the duration time of the longest conversion sequence according to configuration of registers AFEC_MR, AFEC_CHSR, AFEC_SEQ1R, AFEC_SEQ2R.

If a hardware trigger is selected, the start of a conversion is triggered after a delay starting at each rising edge of the selected signal. Due to asynchronous handling, the delay may vary in a range of two peripheral clock periods to one AFE clock period.

Figure 50-6. Conversion Start with the Hardware Trigger



If one of the TIOA outputs is selected, the corresponding Timer Counter channel must be programmed in Waveform mode.

Only one start command is necessary to initiate a conversion sequence on all the channels. The AFEC hardware logic automatically performs the conversions on the active channels, then waits for a new request. The Channel Enable (AFEC_CHER) and Channel Disable (AFEC_CHDR) registers permit the analog channels to be enabled or disabled independently.

If the AFEC is used with a DMA, only the transfers of converted data from enabled channels are performed and the resulting data buffers should be interpreted accordingly.

50.6.7 Sleep Mode and Conversion Sequencer

The AFEC Sleep mode maximizes power saving by automatically deactivating the AFE when it is not being used for conversions. Sleep mode is selected by setting the SLEEP bit in AFEC_MR.

Downloaded from Arrow.com.

Sleep mode is managed by a conversion sequencer, which automatically processes the conversions of all channels at lowest power consumption.

This mode can be used when the minimum period of time between two successive trigger events is greater than the startup period of the AFEC. Refer to the AFE Characteristics in Section 56. "Electrical Characteristics".

When a start conversion request occurs, the AFE is automatically activated. As the analog cell requires a start-up time, the logic waits during this lapse and starts the conversion on the enabled channels. When all conversions are complete, the AFE is deactivated until the next trigger. Triggers occurring during the sequence are not taken into account.

A fast wake-up mode is available in the AFEC_MR as a compromise between power-saving strategy and responsiveness. Setting the FWUP bit enables the Fast Wake-up mode. In Fast Wake-up mode, the AFE is not fully deactivated while no conversion is requested, thereby providing lower power savings but faster wake-up.

The conversion sequencer allows automatic processing with minimum processor intervention and optimized power consumption. Conversion sequences are performed periodically using a Timer/Counter output or the PWM event line.

The DMA can automatically process the periodic acquisition of several samples without processor intervention.

The sequence can be customized by programming the Channel Sequence registers AFEC_SEQ1R and AFEC_SEQ2R and setting the USEQ bit of the AFEC_MR. The user selects a specific order of channels and can program up to 12 conversions by sequence. The user may create a personal sequence by writing channel numbers in AFEC_SEQ1R and AFEC_SEQ2R. Channel numbers can be written in any order and repeated several times. Only enabled USCHx fields are converted. Thus, to program a 15-conversion sequence, the user disables AFEC_CHSR.CH15, thus disabling the field USCH15 of AFEC_SEQ2R.

Note: The reference voltage pins always remain connected in Normal mode as in Sleep mode.

50.6.8 Comparison Window

The AFEC features automatic comparison functions. It compares converted values to a low threshold, a high threshold or both, depending on the value of the CMPMODE bit in AFEC_EMR. The comparison can be done on all channels or only on the channel specified in the CMPSEL field of AFEC_EMR. To compare all channels, the CMPALL bit in AFEC_EMR must be set.

Moreover, a filtering option can be set by writing the number of consecutive comparison errors needed to raise the flag. This number can be written and read in the CMPFILTER field of the AFEC_EMR.

The flag can be read on the COMPE bit of the AFEC_ISR and can trigger an interrupt.

The high threshold and the low threshold can be read/written in the Compare Window Register (AFEC_CWR).

Depending on the sign of the conversion, chosen by setting the SIGNMODE bit in the AFEC Extended Mode Register, the high threshold and low threshold values must be signed or unsigned to maintain consistency during the comparison. If the conversion is signed, both thresholds must also be signed; if the conversion is unsigned, both thresholds must be unsigned. If comparison occurs on all channels, the SIGNMODE bit must be set to ALL_UNSIGNED or ALL_SIGNED and thresholds must be set accordingly.

50.6.9 Differential Inputs

The AFE can be used either as a single-ended AFE (AFEC_DIFFR.DIFF = 0) or as a fully differential AFE (AFEC_DIFFR.DIFF = 1). By default, after a reset, the AFE is in Single-ended mode.

The AFEC can apply a different mode on each channel.

The same inputs are used in Single-ended or Differential mode.

Depending on the AFE mode, the analog multiplexer selects one or two inputs to map to a channel. Table 50-4 provides input mapping for both modes.

Downloaded from Arrow.com.

Table 50-4. Input Pins and Channel Number

Input Pins	Channel Number	
	Single-ended Mode	Differential Mode
AFE_AD0	CH0	CH0
AFE_AD1	CH1	
...	...	...
AFE_AD10	CH10	CH10
AFE_AD11	CH11	

50.6.10 Sample-and-Hold Modes

The AFE can be configured in either single Sample-and-Hold mode (AFEC_SHMR.DUALx = 0) or dual Sample-and-Hold mode (AFEC_SHMR.DUALx = 1). By default, after a reset, the AFE is in single Sample-and-Hold mode.

The AFEC can apply a different mode on each channel.

The same inputs are used in single Sample-and-Hold mode or in dual Sample-and-Hold mode. Single-ended/Differential mode and single/dual Sample-and-Hold mode can be combined. See Table 50-5 and Table 50-6.

Table 50-5. Input Pins and Channel Number In Dual Sample-and-Hold Mode

Single-Ended Input Pins	Differential Input Pins	Channel Numbers
AFE_AD0 & AFE_AD6	AFE_AD0-AD1 & AFE_AD6–AFE_AD7	CH0
AFE_AD1 & AFE_AD7	–	CH1
...	...	...
AFE_AD4 & AFE_AD10	AFE_AD4–AFE_AD5 & AFE_AD10–AFE_AD11	CH4
AFE_AD5 & AFE_AD11	–	CH5

Table 50-6. Input Pins and Channel Number in Single Sample-and-Hold Mode

Single-Ended Input Pins	Differential Input Pins	Channel Number
AFE_AD0	AFE_AD0-AFE_AD1	CH0
AFE_AD1	–	CH1
...	...	...
AFE_AD10	AFE_AD10–AFE_AD11	CH10
AFE_AD11	–	CH11

Downloaded from Arrow.com.

50.6.11 Input Gain and Offset

The AFE has a built-in programmable gain amplifier (PGA) and programmable offset per channel through a DAC.

The programmable gain amplifier can be set to gains of 1, 2 and 4 and can be used for single-ended applications or for fully differential applications.

The AFEC can apply different gain and offset on each channel.

The gain is configured in the GAIN field of the Channel Gain Register (AFEC_CGR) as shown in Table 50-7.

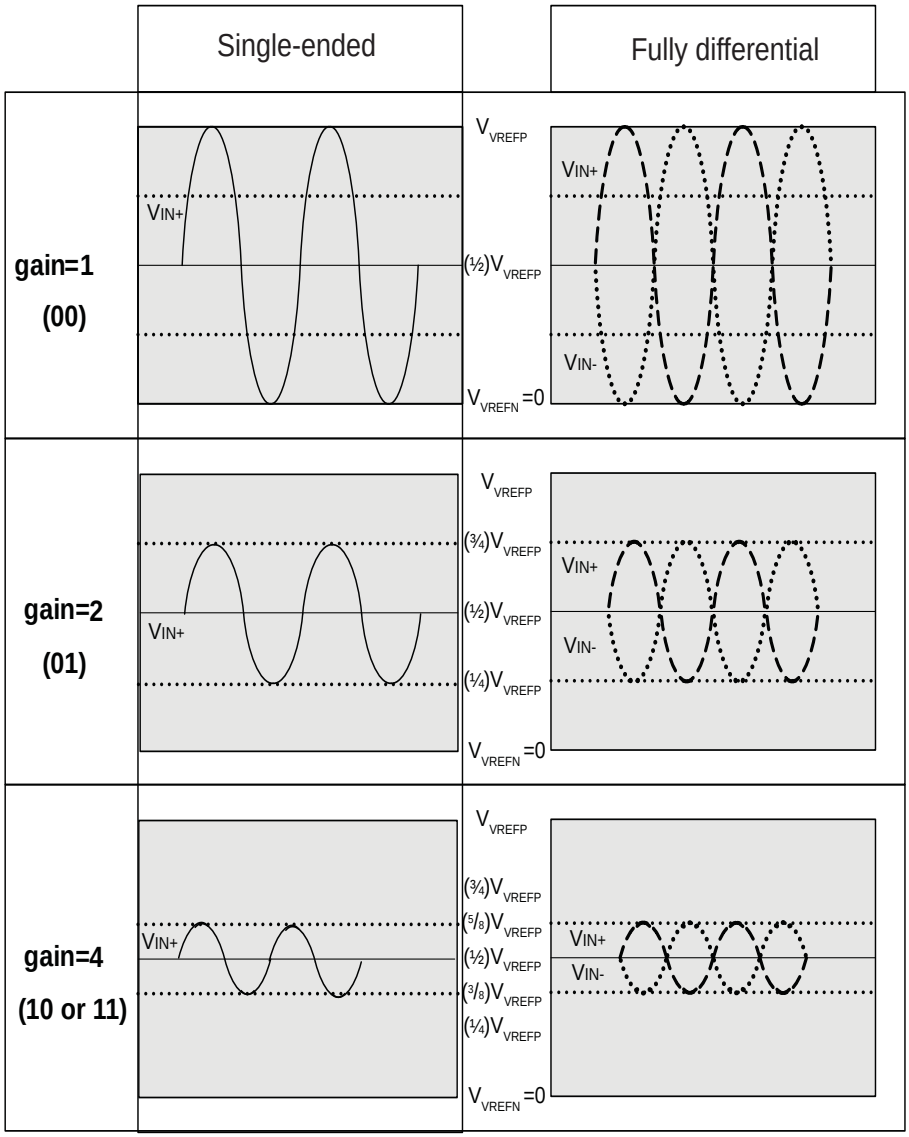
Table 50-7. Gain of the Sample-and-Hold Unit

GAIN	GAIN (DIFFx = 0)	GAIN (DIFFx = 1)
0	1	1
1	2	2
2	4	4
3	4	4

The analog offset of the AFE is configured in the AOFF field in the Channel Offset Compensation register (AFEC_COCR). The offset is only available in Single-ended mode. When AOFF is configured to 0, the offset equals 0; when AOFF is configured to 1023 the offset equals VREFP – 1 LSB. All possible offset values are provided between these two limits according to the following formula: AOFF $\times$ (V_{REF} $\div$ 1024).

Downloaded from Arrow.com.

Figure 50-7. Analog Full Scale Ranges in Single-Ended/Differential Applications Versus Gain



Downloaded from Arrow.com.

50.6.12 AFE Timings

Each AFE has its own minimal startup time configured in the field STARTUP in AFEC_MR.

When the gain, offset or differential input parameters of the analog cell change between two channels, the analog cell may need a specific settling time before starting the tracking phase. In this case, the controller waits during the settling time defined in the AFEC_MR.

Warning: No input buffer amplifier to isolate the source is included in the AFE. This must be taken into consideration.

50.6.13 Temperature Sensor

The temperature sensor is internally connected to channel index 11.

The AFEC manages temperature measurement in several ways. The different methods of measurement depend on the configuration bits TRGEN in the AFEC_MR and CH11 in AFEC_CHSR.

Temperature measurement can be triggered at the same rate as other channels by enabling the conversion channel 11.

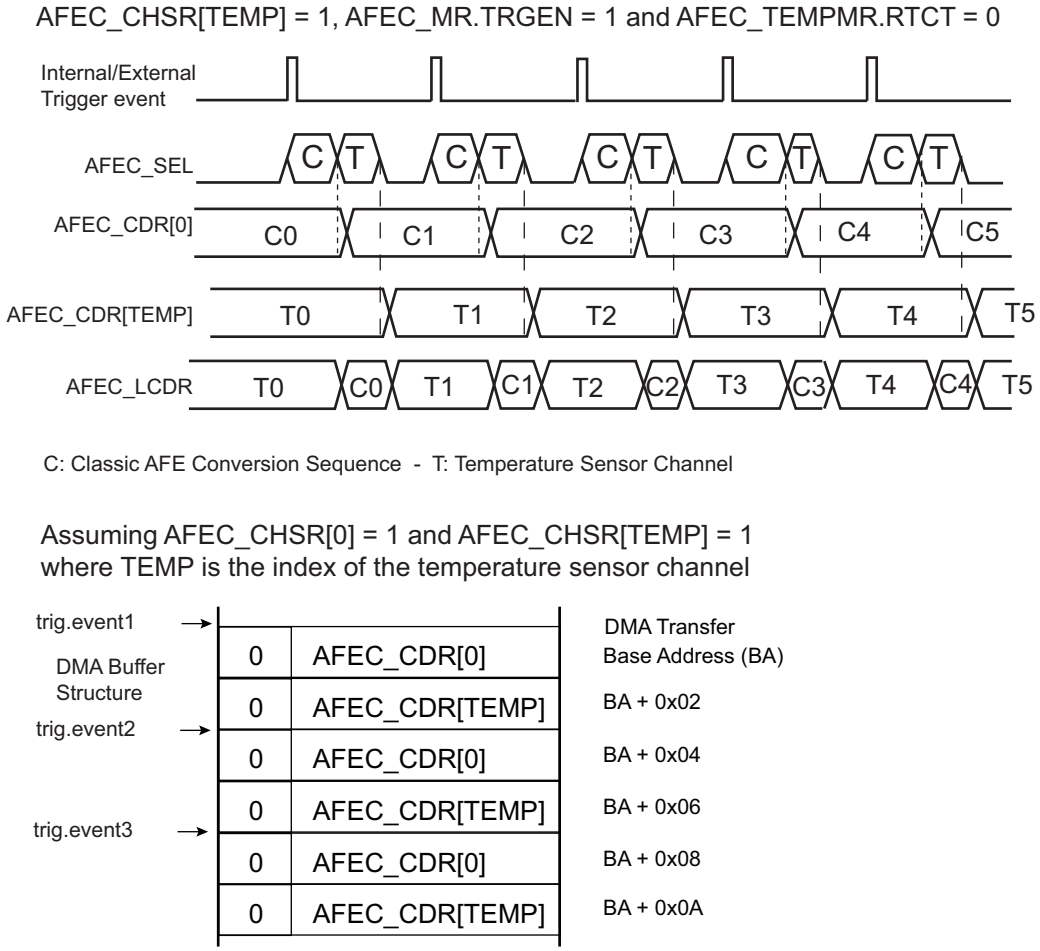
If the bit CH11 in AFEC_CHSR is enabled, the temperature sensor analog cell is switched on. If a user sequence is used, the last converted channel of the sequence is always the temperature sensor channel.

A manual start can be performed only if TRGEN bit in AFEC_MR is disabled. When the START bit in AFEC_CR is set, the temperature sensor channel conversion is scheduled together with the other enabled channels (if any). The result of the conversion is placed in an internal register that can be read in the AFEC_CDR (AFEC_CSELR must be programmed accordingly prior to reading AFEC_CDR) and the associated flag EOC11 is set in the AFEC_ISR.

The channel of the temperature sensor is periodically converted together with the other enabled channels and the result is placed into AFEC_LCDR and an internal register (can be read in AFEC_CDR). Thus the temperature conversion result is part of the Peripheral DMA Controller buffer. The temperature channel can be enabled/disabled at anytime, but this may not be optimal for downstream processing.

Downloaded from Arrow.com.

Figure 50-8. Non-Optimized Temperature Conversion



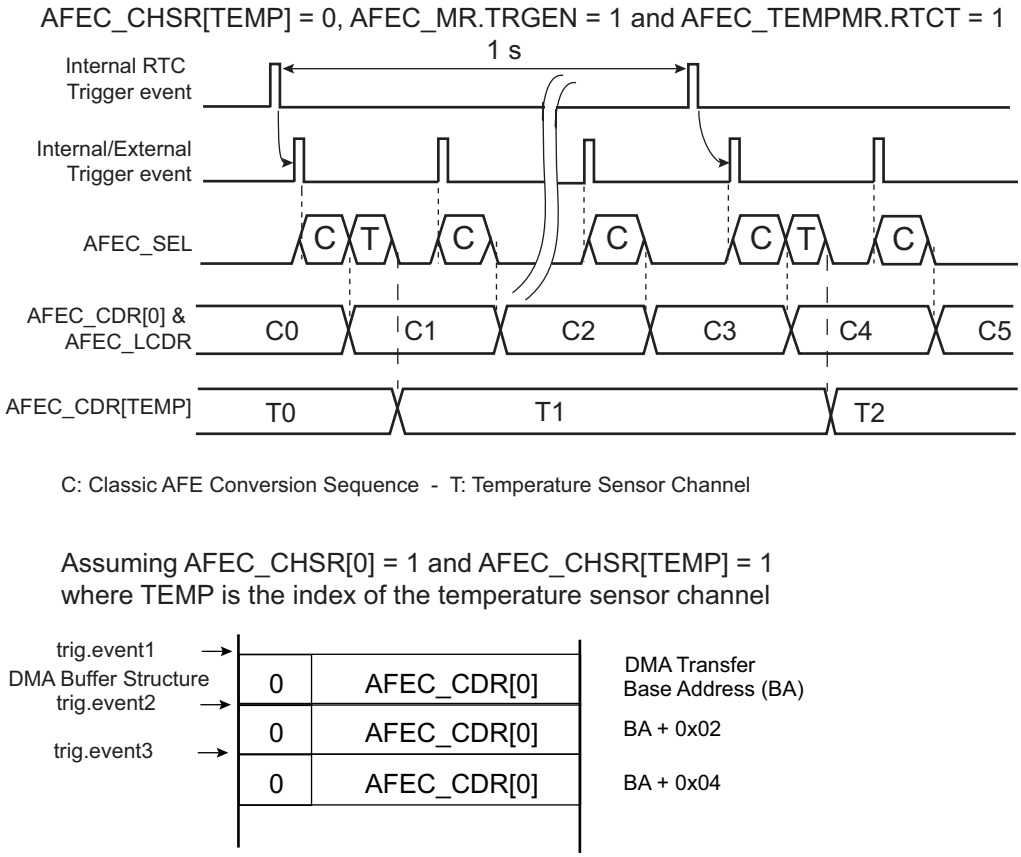
The temperature factor has a slow variation rate and may be different from other conversion channels. As a result, the AFEC allows a different way of triggering temperature measurement when the bit RTCT is set in the AFEC_TEMPMR but the CH11 is cleared in the AFEC_CHSR.

In this configuration, the measurement is triggered every second by means of an internal trigger generated by the RTC. This trigger is always enabled and independent of the triggers used for other channels. It is selected in the TRGSEL field in AFEC_MR. In this mode of operation, the temperature sensor is only powered for a period of time covering startup time and conversion time.

Every second, a conversion is scheduled for channel 11 but the result of the conversion is only uploaded to an internal register read by means of AFEC_CDR, and not to AFEC_LCDR. Therefore, the temperature channel is not part of the Peripheral DMA Controller buffer; only the enabled channel are kept in the buffer. The end of conversion of the temperature channel is reported by means of the EOC11 flag in AFEC_ISR.

Downloaded from Arrow.com.

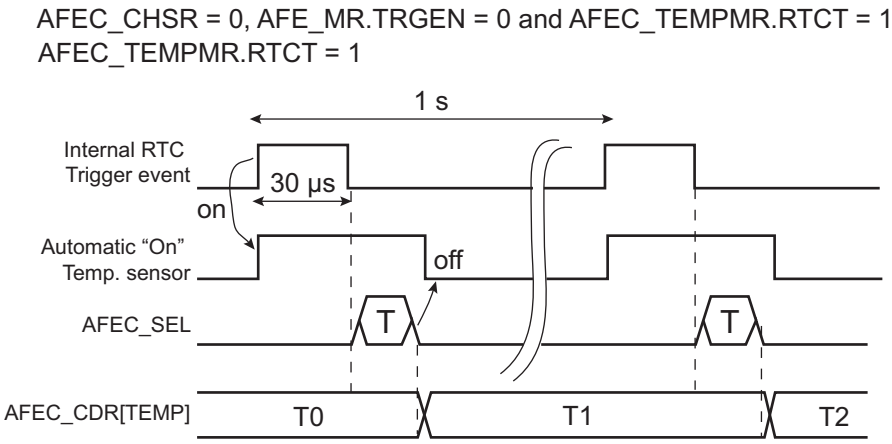
Figure 50-9. Optimized Temperature Conversion Combined with Classical Conversions



If RTCT is set and TRGEN is cleared, then all channels are disabled (AFEC_CHSR = 0) and only channel 11 is converted at a rate of one conversion per second.

This mode of operation, when combined with Sleep mode operation, provides a low-power mode for temperature measurement assuming there is no other AFE conversion to schedule at a higher sampling rate or no other channel to convert.

Figure 50-10. Temperature Conversion Only



Moreover, it is possible to raise a flag only if there is predefined change in the temperature measurement. The user can define a range of temperature or a threshold in AFEC_TEMPCWR and the mode of comparison in AFEC_TEMPMR. These values define the way the TEMPCHG flag will be raised in AFEC_ISR.

Downloaded from Arrow.com.

The TEMPCHG flag can be used to trigger an interrupt if there is an update/modification to be made in the system resulting from a temperature change.

In any case, if temperature sensor measurement is configured, the temperature can be read at anytime in AFEC_CDR (AFEC_CSELR must be programmed accordingly prior to reading AFEC_CDR) .

50.6.14 Enhanced Resolution Mode and Digital Averaging Function

The Enhanced Resolution mode is enabled when the field RES is set to 13-bit resolution or higher in AFEC_EMR. In this mode, the AFEC trades conversion performance for accuracy by averaging multiple samples, thus providing a digital low-pass filter function. The resolution mode selected determines the oversampling, which represents the performance reduction factor.

To increase the accuracy by averaging multiple samples, some noise must be present in the input signal. The noise level should be between one and two LSB peak-to-peak to get good averaging performance.

Table 50-8 summarizes the oversampling ratio depending on the resolution mode selected.

Table 50-8. Resolution and Oversampling Ratio

Resolution Mode	Oversampling Ratio
13-bit	4
14-bit	16
15-bit	64
16-bit	256

Free Run mode is not supported if Enhanced Resolution mode is used.

The selected oversampling ratio applies to all enabled channels except the temperature sensor channel if triggered by an RTC event. See Section 50.6.13 ”Temperature Sensor”.

The average result is valid into an internal register (read by means of the AFEC_CDR) only if EOCx (x corresponding to the index of the channel) flag is set in AFEC_ISR and OVREx flag is cleared in the AFEC_OVER. The average result is valid for all channels in the AFEC_LCDR only if DRDY is set and GOVRE is cleared in the AFEC_ISR.

Note that the AFEC_CDR is not buffered. Therefore, when an averaging sequence is on- going, the value in this register changes after each averaging sample. However, overrun flags in the AFEC_OVER rise as soon as the first sample of an averaging sequence is received. Thus the previous averaged value is not read, even if the new averaged value is not ready.

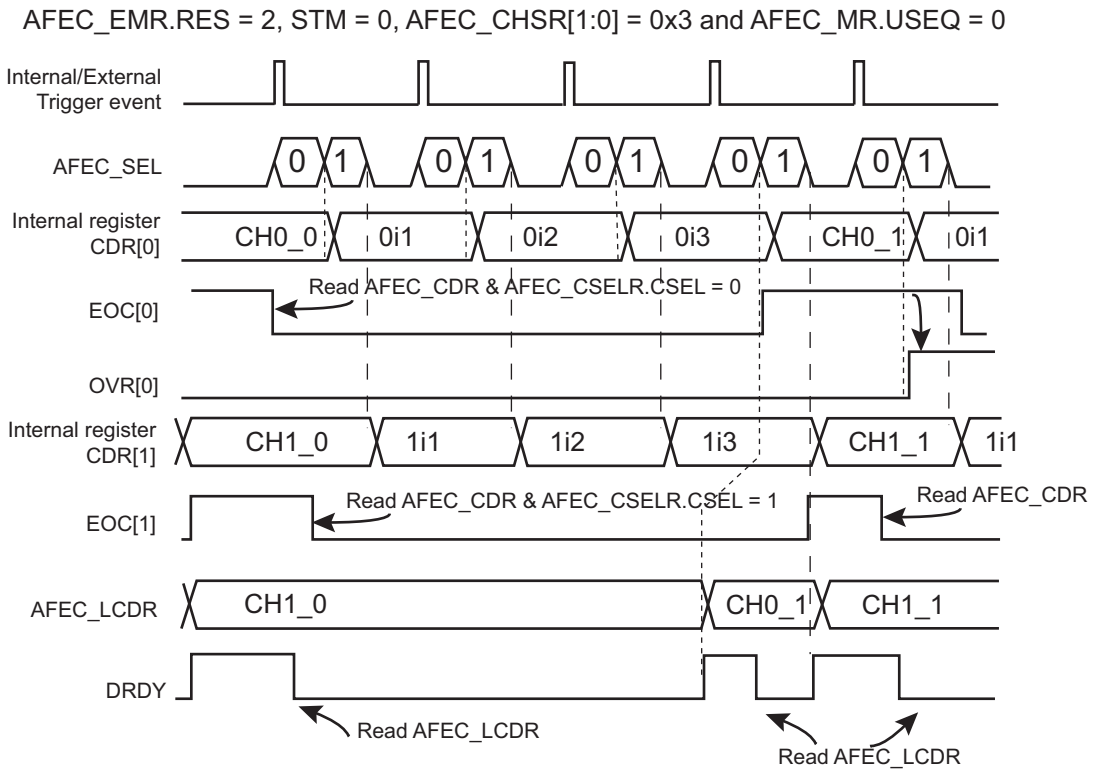
As a result, when an overrun flag rises in the AFEC_OVER, this indicates only that the previous unread data is lost. It does not indicate that this data has been overwritten by the new averaged value, as the averaging sequence concerning this channel can still be on-going.

The samples can be defined in different ways for the averaging function depending on the configuration of the STM bit in AFEC_EMR and the USEQ bit in AFEC_MR.

When USEQ is cleared, there are two possible ways to generate the averaging through the trigger event. If the STM bit is cleared in AFEC_EMR, every trigger event generates one sample for each enabled channel, as described in Figure 50-11. Therefore, four trigger events are requested to get the result of averaging if RES = 2.

Downloaded from Arrow.com.

Figure 50-11. Digital Averaging Function Waveforms over Multiple Trigger Events

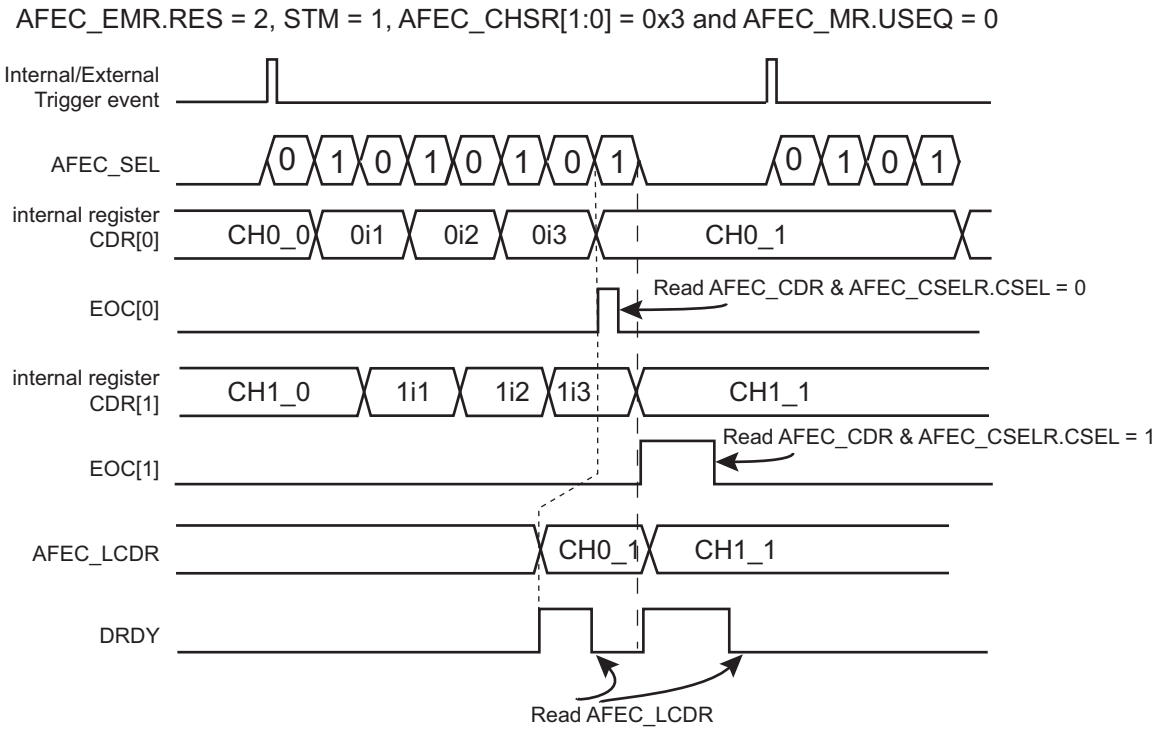


Note: 0i1,0i2,0i3, 1i1, 1i2, 1i3 are intermediate results and CH0/1_0/1 are final result of average function.

Downloaded from Arrow.com.

If the STM bit is set in AFEC_EMR and the USEQ bit is cleared in AFEC_MR, the sequence to be converted, defined in the AFEC_CHSR, is automatically repeated n times, where n corresponds to the oversampling ratio defined in the RES field in AFEC_EMR. As a result, only one trigger is required to get the result of the averaging function as shown in Figure 50-12.

Figure 50-12. Digital Averaging Function Waveforms on a Single Trigger Event



Note: 0i1, 0i2, 0i3, 1i1, 1i2, 1i3 are intermediate results and CH0/1_0/1 are final result of average function.

Downloaded from Arrow.com.

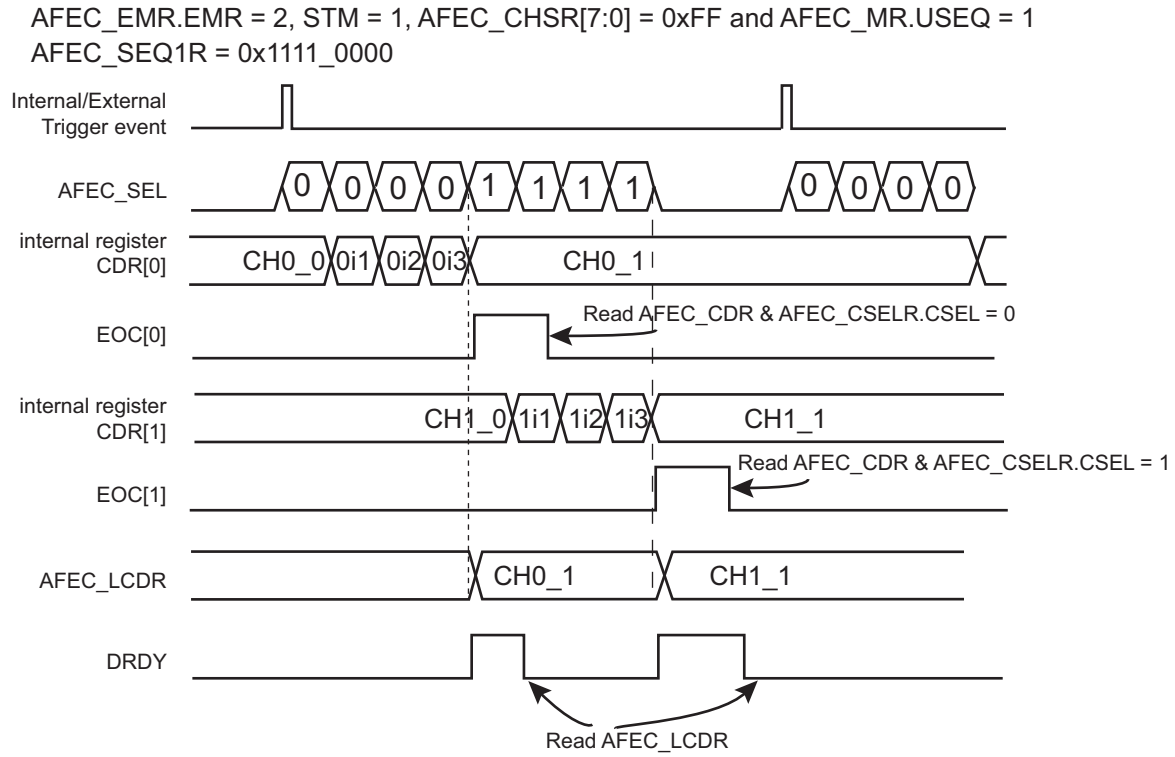
When USEQ is set, the user can define the channel sequence to be converted by configuring AFEC_SEQxR and AFEC_CHER so that channels are not interleaved during the averaging period. Under these conditions, a sample is defined for each end of conversion as described in Figure 50-13.

Therefore, if the same channel is configured to be converted four times consecutively and RES = 2 in the AFEC_EMR, the averaging result is placed in the corresponding channel internal data register (read by means of the AFEC_CDR) and the AFEC_LCDR for each trigger event.

In this case, the AFE real sample rate remains the maximum AFE sample rate divided by 4.

When USEQ is set and the RES field enables the Enhanced Resolution mode, it is important to note that the user sequence must be a sequence being an integer multiple of 4 (i.e., the number of the enabled channel in the Channel Status register (AFEC_CHSR) must be an integer multiple of 4 and the AFEC_SEQxR must be a series of 4 times the same channel index).

Figure 50-13. Digital Averaging Function Waveforms on a Single Trigger Event, Non-interleaved



Note: 0i1, 0i2, 0i3, 1i1, 1i2, 1i3 are intermediate results and CH0/1_0/1 are final result of average function.

Downloaded from Arrow.com.

50.6.15 Automatic Error Correction

The AFEC features automatic error correction of conversion results. Offset and gain error corrections are available. The correction can be enabled for each channel and correction values (offset and gain) are defined per Sample & Hold unit.

To enable error correction, the ECORR bit must be set in the AFEC Channel Error Correction Register (AFEC_CECR). The offset and gain values used to compensate the results are set per Sample & Hold unit basis using the AFEC Correction Select Register (AFEC_COSR) and the AFEC Correction Values Register (AFEC_CVR). The selection register (AFEC_COSR) is used to select the Sample & Hold unit to be displayed in AFEC_CVR This selection applies both to read and write operations in AFEC_CVR.

The final conversion result after error correction is obtained using the following formula, with:

- OFFSETCORR—the Offset Correction value. OFFSETCORR is a signed value.
- GAINCORR—the Gain Correction value
- Gs—the size of the GAINCORR field, equal to 11
- ConvValue—the value converted by the AFE (as returned in AFEC_LCDR or AFEC_CDR)
- Resolution—the resolution used to process the conversion (either 12 13, 14, 15, or 16).

$$CorrectedData = (ConvertedData + OFFSETCORR) \times \frac{GAINCORR}{2^{(Gs-1)}}$$

Downloaded from Arrow.com.

50.6.16 Buffer Structure

The DMA read channel is triggered each time a new data is stored in AFEC_LCDR. The same structure of data is repeatedly stored in AFEC_LCDR each time a trigger event occurs. Depending on the user mode of operation (AFEC_MR, AFEC_CHSR, AFEC_SEQ1R, AFEC_SEQ2R) the structure differs. When TAG is cleared, each data transferred to DMA buffer is carried on a half-word (16-bit) and consists of the last converted data right-aligned. When TAG is set, this data is carried on a word buffer (32-bit) and CHNB carries the channel number, thus simplifying post-processing in the DMA buffer and ensuring the integrity of the DMA buffer.

50.6.17 Fault Output

The AFEC internal fault output is directly connected to PWM fault input. Fault output may be asserted depending on the configuration of AFEC_EMR and AFEC_CWR and converted values. When the compare occurs, the AFEC fault output generates a pulse of one peripheral clock cycle to the PWM fault input. This fault line can be enabled or disabled within the PWM. If it is activated and asserted by the AFEC, the PWM outputs are immediately placed in a safe state (pure combinational path). Note that the AFEC fault output connected to the PWM is not the COMPE bit. Thus the Fault Mode (FMOD) within the PWM configuration must be FMOD = 1.

Downloaded from Arrow.com.

50.6.18 Register Write Protection

To prevent any single software error from corrupting AFEC behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the AFEC Write Protection Mode Register (AFEC_WPMR).

If a write access to a write-protected register is detected, the WPVS flag in the AFEC Write Protection Status Register (AFEC_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS flag is automatically cleared by reading the AFEC_WPSR.

The protected registers are:

- AFEC Mode Register
- AFEC Extended Mode Register
- AFEC Channel Sequence 1 Register
- AFEC Channel Sequence 2 Register
- AFEC Channel Enable Register
- AFEC Channel Disable Register
- AFEC Compare Window Register
- AFEC Channel Gain Register
- AFEC Channel Differential Register
- AFEC Channel Selection Register
- AFEC Channel Offset Compensation Register
- AFEC Temperature Sensor Mode Register
- AFEC Temperature Compare Window Register
- AFEC Analog Control Register
- AFEC Sample & Hold Mode Register
- AFEC Correction Select Register
- AFEC Correction Values Register
- AFEC Channel Error Correction Register

Downloaded from Arrow.com.

50.7 Analog Front-End Controller (AFEC) User Interface

Table 50-9. Register Mapping

Offset[1]	Register	Name	Access	Reset
0x00	AFEC Control Register	AFEC_CR	Write-only	–
0x04	AFEC Mode Register	AFEC_MR	Read/Write	0x00000000
0x08	AFEC Extended Mode Register	AFEC_EMR	Read/Write	0x00000000
0x0C	AFEC Channel Sequence 1 Register	AFEC_SEQ1R	Read/Write	0x00000000
0x10	AFEC Channel Sequence 2 Register	AFEC_SEQ2R	Read/Write	0x00000000
0x14	AFEC Channel Enable Register	AFEC_CHER	Write-only	–
0x18	AFEC Channel Disable Register	AFEC_CHDR	Write-only	–
0x1C	AFEC Channel Status Register	AFEC_CHSR	Read-only	0x00000000
0x20	AFEC Last Converted Data Register	AFEC_LCDR	Read-only	0x00000000
0x24	AFEC Interrupt Enable Register	AFEC_IER	Write-only	–
0x28	AFEC Interrupt Disable Register	AFEC_IDR	Write-only	–
0x2C	AFEC Interrupt Mask Register	AFEC_IMR	Read-only	0x00000000
0x30	AFEC Interrupt Status Register	AFEC_ISR	Read-only	0x00000000
0x34–0x40	Reserved	–	–	–
0x44–0x48	Reserved	–	–	–
0x4C	AFEC Overrun Status Register	AFEC_OVER	Read-only	0x00000000
0x50	AFEC Compare Window Register	AFEC_CWR	Read/Write	0x00000000
0x54	AFEC Channel Gain Register	AFEC_CGR	Read/Write	0x00000000
0x5C	Reserved	–	–	–
0x60	AFEC Channel Differential Register	AFEC_DIFFR	Read/Write	0x00000000
0x64	AFEC Channel Selection Register	AFEC_CSELR	Read/Write	0x00000000
0x68	AFEC Channel Data Register	AFEC_CDR	Read-only	0x00000000
0x6C	AFEC Channel Offset Compensation Register	AFEC_COCR	Read/Write	0x00000000
0x70	AFEC Temperature Sensor Mode Register	AFEC_TEMPMR	Read/Write	0x00000000
0x74	AFEC Temperature Compare Window Register	AFEC_TEMPCWR	Read/Write	0x00000000
0x78–0x90	Reserved	–	–	–
0x94	AFEC Analog Control Register	AFEC_ACR	Read/Write	0x00000100
0x98–0x9C	Reserved	–	–	–
0xA0	AFEC Sample & Hold Mode Register	AFEC_SHMR	Read/Write	0x00000000
0xA4–0xAC	Reserved	–	–	–
0xD0	AFEC Correction Select Register	AFEC_COSR	Read/Write	0x00000000
0xD4	AFEC Correction Values Register	AFEC_CVR	Read/Write	0x00000000
0xD8	AFEC Channel Error Correction Register	AFEC_CECR	Read/Write	0x00000000
0xDC–0xE0	Reserved	–	–	–

Downloaded from Arrow.com.

Table 50-9. Register Mapping (Continued)

Offset[1]	Register	Name	Access	Reset
0xE4	AFEC Write Protection Mode Register	AFEC_WPMR	Read/Write	0x00000000
0xE8	AFEC Write Protection Status Register	AFEC_WPSR	Read-only	0x00000000
0xEC–0xF8	Reserved	–	–	–
0xFC	Reserved	–	–	–

Notes: 1. Any offset not listed in Table 50-9 must be considered as “reserved”.

Downloaded from Arrow.com.

50.7.1 AFEC Control Register

Name: AFEC_CR

Address: 0x4003C000 (0), 0x40064000 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	START	SWRST

- **SWRST: Software Reset**

0: No effect.

1: Resets the AFEC simulating a hardware reset.

- **START: Start Conversion**

0: No effect.

1: Begins Analog Front-End conversion.

Downloaded from Arrow.com.

50.7.2 AFEC Mode Register

Name: AFEC_MR

Address: 0x4003C004 (0), 0x40064004 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
USEQ	–	TRANSFER		TRACKTIM			
23	22	21	20	19	18	17	16
ONE	–	–		STARTUP			
15	14	13	12	11	10	9	8
PRESCAL							
7	6	5	4	3	2	1	0
FREERUN	FWUP	SLEEP	–	TRGSEL			TRGEN

This register can only be written if the WPEN bit is cleared in the AFEC Write Protection Mode Register.

- **TRGEN: Trigger Enable**

Value	Name	Description
0	DIS	Hardware triggers are disabled. Starting a conversion is only possible by software.
1	EN	Hardware trigger selected by TRGSEL field is enabled.

- **TRGSEL: Trigger Selection**

Value	Name	Description
0	AFEC_TRIG0	AFE0_ADTRG for AFEC0 / AFE1_ADTRG for AFEC1
1	AFEC_TRIG1	TIOA Output of the Timer Counter Channel 0 for AFEC0/TIOA Output of the Timer Counter Channel 3 for AFEC1
2	AFEC_TRIG2	TIOA Output of the Timer Counter Channel 1 for AFEC0/TIOA Output of the Timer Counter Channel 4 for AFEC1
3	AFEC_TRIG3	TIOA Output of the Timer Counter Channel 2 for AFEC0/TIOA Output of the Timer Counter Channel 5 for AFEC1
4	AFEC_TRIG4	PWM0 event line 0 for AFEC0 / PWM1 event line 0 for AFEC1
5	AFEC_TRIG5	PWM0 event line 1 for AFEC0 / PWM1 event line 1 for AFEC1
6	AFEC_TRIG6	Analog Comparator
7	–	Reserved

- **SLEEP: Sleep Mode**

Value	Name	Description
0	NORMAL	Normal mode: The AFE and reference voltage circuitry are kept ON between conversions.
1	SLEEP	Sleep mode: The AFE and reference voltage circuitry are OFF between conversions.

- **FWUP: Fast Wake-up**

Value	Name	Description
0	OFF	Normal Sleep mode: The sleep mode is defined by the SLEEP bit.
1	ON	Fast wake-up Sleep mode: The voltage reference is ON between conversions and AFE is OFF.

Downloaded from Arrow.com.

- **FREERUN: Free Run Mode**

Value	Name	Description
0	OFF	Normal mode
1	ON	Free Run mode: Never wait for any trigger.

- **PRESCAL: Prescaler Rate Selection**

PRESCAL = $f_{peripheral\ clock}/ f_{AFE\ Clock}$ - 1

When PRESCAL is cleared, no conversion is performed.

- **STARTUP: Start-up Time**

Value	Name	Description
0	SUT0	0 periods of AFE clock
1	SUT8	8 periods of AFE clock
2	SUT16	16 periods of AFE clock
3	SUT24	24 periods of AFE clock
4	SUT64	64 periods of AFE clock
5	SUT80	80 periods of AFE clock
6	SUT96	96 periods of AFE clock
7	SUT112	112 periods of AFE clock
8	SUT512	512 periods of AFE clock
9	SUT576	576 periods of AFE clock
10	SUT640	640 periods of AFE clock
11	SUT704	704 periods of AFE clock
12	SUT768	768 periods of AFE clock
13	SUT832	832 periods of AFE clock
14	SUT896	896 periods of AFE clock
15	SUT960	960 periods of AFE clock

- **ONE: One**

This bit must be written to 1.

- **TRACKTIM: Tracking Time**

Inherent tracking time is always 15 AFE clock cycles.

- **TRANSFER: Transfer Period**

The TRANSFER field should be set to 0 to guarantee the optimal transfer time.

- **USEQ: User Sequence Enable**

Value	Name	Description
0	NUM_ORDER	Normal mode: The controller converts channels in a simple numeric order.
1	REG_ORDER	User Sequence mode: The sequence respects what is defined in AFEC_SEQ1R and AFEC_SEQ1R.

Downloaded from Arrow.com.

50.7.3 AFEC Extended Mode Register

Name: AFEC_EMR

Address: 0x4003C008 (0), 0x40064008 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	SIGNMODE		–	–	STM	TAG
23	22	21	20	19	18	17	16
–	–	–	–	–	RES		
15	14	13	12	11	10	9	8
–	–	CMPFILTER		–	–	CMPALL	–
7	6	5	4	3	2	1	0
CMPSEL					–	CMPMODE	

This register can only be written if the WPEN bit is cleared in the AFEC Write Protection Mode Register.

- **CMPMODE: Comparison Mode**

Value	Name	Description
0	LOW	Generates an event when the converted data is lower than the low threshold of the window.
1	HIGH	Generates an event when the converted data is higher than the high threshold of the window.
2	IN	Generates an event when the converted data is in the comparison window.
3	OUT	Generates an event when the converted data is out of the comparison window.

- **CMPSEL: Comparison Selected Channel**

If CMPALL = 0: CMPSEL indicates which channel has to be compared.

If CMPALL = 1: No effect.

- **CMPALL: Compare All Channels**

0: Only the channel indicated in CMPSEL field is compared.

1: All channels are compared.

- **CMPFILTER: Compare Event Filtering**

Number of consecutive compare events necessary to raise the flag = CMPFILTER+1.

When programmed to ‘0’, the flag rises as soon as an event occurs.

- **RES: Resolution**

Value	Name	Description
0	NO_AVERAGE	12-bit resolution, AFE sample rate is maximum (no averaging).
2	OSR4	13-bit resolution, AFE sample rate divided by 4 (averaging).
3	OSR16	14-bit resolution, AFE sample rate divided by 16 (averaging).
4	OSR64	15-bit resolution, AFE sample rate divided by 64 (averaging).
5	OSR256	16-bit resolution, AFE sample rate divided by 256 (averaging).

Downloaded from Arrow.com.

- **TAG: TAG of the AFEC_LDCR**

0: Clears CHNB in AFEC_LDCR.

1: Appends the channel number to the conversion result in AFEC_LDCR.

- **STM: Single Trigger Mode**

0: Multiple triggers are required to get an averaged result.

1: Only a single trigger is required to get an averaged value.

- **SIGNMODE: Sign Mode**

Value	Name	Description
0	SE_UNSG_DF_SIGN	Single-Ended channels: Unsigned conversions. Differential channels: Signed conversions.
1	SE_SIGN_DF_UNSG	Single-Ended channels: Signed conversions. Differential channels: Unsigned conversions.
2	ALL_UNSIGNED	All channels: Unsigned conversions.
3	ALL_SIGNED	All channels: Signed conversions.

Note: If conversion results are signed and resolution is below 16 bits, the sign is extended up to the bit 15 (for example, 0xF43 for 12-bit resolution will be read as 0xFF43 and 0x467 will be read as 0x0467). See Section 50.6.5 ”Conversion Results Format”.

Downloaded from Arrow.com.

50.7.4 AFEC Channel Sequence 1 Register

Name: AFEC_SEQ1R

Address: 0x4003C00C (0), 0x4006400C (1)

Access: Read/Write

31	30	29	28	27	26	25	24
USCH7				USCH6			
23	22	21	20	19	18	17	16
USCH5				USCH4			
15	14	13	12	11	10	9	8
USCH3				USCH2			
7	6	5	4	3	2	1	0
USCH1				USCH0			

This register can only be written if the WPEN bit is cleared in the AFEC Write Protection Mode Register.

- **USCHx: User Sequence Number x**

The sequence number x (USCHx) can be programmed by the Channel number CHy where y is the value written in this field. The allowed range is 0 up to 11. So it is only possible to use the sequencer from CH0 to CH11.

This register activates only if AFEC_MR.USEQ bit is set.

Any USCHx field is taken into account only if the AFEC_CHSR.CHx bit is set, else any value written in USCHx does not add the corresponding channel in the conversion sequence.

Configuring the same value in different fields leads to multiple samples of the same channel during the conversion sequence. This can be done consecutively, or not, depending on user needs.

Downloaded from Arrow.com.

50.7.5 AFEC Channel Sequence 2 Register

Name: AFEC_SEQ2R

Address: 0x4003C010 (0), 0x40064010 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
USCH11				USCH10			
7	6	5	4	3	2	1	0
USCH9				USCH8			

This register can only be written if the WPEN bit is cleared in the AFEC Write Protection Mode Register.

- **USCHx: User Sequence Number x**

The sequence number x (USCHx) can be programmed by the Channel number CHy where y is the value written in this field. The allowed range is 0 up to 11. So it is only possible to use the sequencer from CH0 to CH11.

This register activates only if AFEC_MR(USEQ) field is set.

Any USCHx field is taken into account only if the AFEC_CHSR.CHx bit is written to one, else any value written in USCHx does not add the corresponding channel in the conversion sequence.

Configuring the same value in different fields leads to multiple samples of the same channel during the conversion sequence. This can be done consecutively, or not, according to user needs.

Downloaded from Arrow.com.

50.7.6 AFEC Channel Enable Register

Name: AFEC_CHER

Address: 0x4003C014 (0), 0x40064014 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	CH11	CH10	CH9	CH8
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

This register can only be written if the WPEN bit is cleared in the AFEC Write Protection Mode Register.

- **CHx: Channel x Enable**

0: No effect.

1: Enables the corresponding channel.

Note: If USEQ = 1 in the AFEC_MR, CHx corresponds to the xth channel of the sequence described in AFEC_SEQ1R, AFEC_SEQ2R.

Downloaded from Arrow.com.

50.7.7 AFEC Channel Disable Register

Name: AFEC_CHDR

Address: 0x4003C018 (0), 0x40064018 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	CH11	CH10	CH9	CH8
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

This register can only be written if the WPEN bit is cleared in the AFEC Write Protection Mode Register.

- **CHx: Channel x Disable**

0: No effect.

1: Disables the corresponding channel.

Warning: If the corresponding channel is disabled during a conversion or if it is disabled and then reenabled during a conversion, its associated data and its corresponding EOCx and GOVRE flags in AFEC_ISR and OVREx flags in AFEC_OVER are unpredictable.

Downloaded from Arrow.com.

50.7.8 AFEC Channel Status Register

Name: AFEC_CHSR

Address: 0x4003C01C (0), 0x4006401C (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	CH11	CH10	CH9	CH8
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHx: Channel x Status**

0: The corresponding channel is disabled.

1: The corresponding channel is enabled.

Downloaded from Arrow.com.

50.7.9 AFEC Last Converted Data Register

Name: AFEC_LCDR

Address: 0x4003C020 (0), 0x40064020 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	CHNB			
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
LDATA							
7	6	5	4	3	2	1	0
LDATA							

- **LDATA: Last Data Converted**

The AFE conversion data is placed into this register at the end of a conversion and remains until a new conversion is completed.

- **CHNB: Channel Number**

Indicates the last converted channel when TAG is set in the AFEC_EMR. If TAG is cleared, CHNB = 0.

Downloaded from Arrow.com.

50.7.10 AFEC Interrupt Enable Register

Name: AFEC_IER

Address: 0x4003C024 (0), 0x40064024 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	TEMPCHG	–	–	–	COMPE	GOVRE	DRDY
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	EOC11	EOC10	EOC9	EOC8
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

- **EOCx: End of Conversion Interrupt Enable x**
- **DRDY: Data Ready Interrupt Enable**
- **GOVRE: General Overrun Error Interrupt Enable**
- **COMPE: Comparison Event Interrupt Enable**
- **TEMPCHG: Temperature Change Interrupt Enable**

Downloaded from Arrow.com.

50.7.11 AFEC Interrupt Disable Register

Name: AFEC_IDR

Address: 0x4003C028 (0), 0x40064028 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	TEMPCHG	–	–	–	COMPE	GOVRE	DRDY
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	EOC11	EOC10	EOC9	EOC8
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

- **EOCx: End of Conversion Interrupt Disable x**
- **DRDY: Data Ready Interrupt Disable**
- **GOVRE: General Overrun Error Interrupt Disable**
- **COMPE: Comparison Event Interrupt Disable**
- **TEMPCHG: Temperature Change Interrupt Disable**

Downloaded from Arrow.com.

50.7.12 AFEC Interrupt Mask Register

Name: AFEC_IMR

Address: 0x4003C02C (0), 0x4006402C (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	TEMPCHG	–	–	–	COMPE	GOVRE	DRDY
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	EOC11	EOC10	EOC9	EOC8
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

- **EOCx: End of Conversion Interrupt Mask x**
- **DRDY: Data Ready Interrupt Mask**
- **GOVRE: General Overrun Error Interrupt Mask**
- **COMPE: Comparison Event Interrupt Mask**
- **TEMPCHG: Temperature Change Interrupt Mask**

Downloaded from Arrow.com.

50.7.13 AFEC Interrupt Status Register

Name: AFEC_ISR

Address: 0x4003C030 (0), 0x40064030 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	TEMPCHG	–	–	–	COMPE	GOVRE	DRDY
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	EOC11	EOC10	EOC9	EOC8
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion x (cleared by reading AFEC_CDRx)**

0: The corresponding analog channel is disabled, or the conversion is not finished. This flag is cleared when reading the AFEC_CDR if the CSEL bit is programmed with ‘x’ in the AFEC_CSELR.

1: The corresponding analog channel is enabled and conversion is complete.

- **TEMPCHG: Temperature Change (cleared on read)**

0: There is no comparison match (defined in the AFEC_TEMPCMPR) since the last read of AFEC_ISR.

1: The temperature value reported on AFEC_CDR (AFEC_CSELR.CSEL = 11) has changed since the last read of AFEC_ISR, according to what is defined in the Temperature Mode register (AFEC_TEMPMR) and the Temperature Compare Window register (AFEC_TEMPCWR).

- **DRDY: Data Ready (cleared by reading AFEC_LCDR)**

0: No data has been converted since the last read of AFEC_LCDR.

1: At least one data has been converted and is available in AFEC_LCDR.

- **GOVRE: General Overrun Error (cleared by reading AFEC_ISR)**

0: No general overrun error occurred since the last read of AFEC_ISR.

1: At least one general overrun error has occurred since the last read of AFEC_ISR.

- **COMPE: Comparison Error (cleared by reading AFEC_ISR)**

0: No comparison error since the last read of AFEC_ISR.

1: At least one comparison error has occurred since the last read of AFEC_ISR.

Downloaded from Arrow.com.

50.7.14 AFEC Overrun Status Register

Name: AFEC_OVER

Address: 0x4003C04C (0), 0x4006404C (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	OVRE11	OVRE10	OVRE9	OVRE8
7	6	5	4	3	2	1	0
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0

- **OVREx: Overrun Error x**

0: No overrun error on the corresponding channel since the last read of AFEC_OVER.

1: There has been an overrun error on the corresponding channel since the last read of AFEC_OVER.

Note: An overrun error does not always mean that the unread data has been replaced by a new valid data. Refer to Section 50.6.14 "Enhanced Resolution Mode and Digital Averaging Function" for details.

Downloaded from Arrow.com.

50.7.15 AFEC Compare Window Register

Name: AFEC_CWR

Address: 0x4003C050 (0), 0x40064050 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
HIGHTHRES							
23	22	21	20	19	18	17	16
HIGHTHRES							
15	14	13	12	11	10	9	8
LOWTHRES							
7	6	5	4	3	2	1	0
LOWTHRES							

This register can only be written if the WPEN bit is cleared in the AFEC Write Protection Mode Register.

- **LOWTHRES: Low Threshold**

Low threshold associated to compare settings of AFEC_EMR. For comparisons lower than 16 bits and signed, the sign should be extended up to the bit 15.

- **HIGHTHRES: High Threshold**

High threshold associated to compare settings of AFEC_EMR. For comparisons lower than 16 bits and signed, the sign should be extended up to the bit 15.

Downloaded from Arrow.com.

50.7.16 AFEC Channel Gain Register

Name: AFEC_CGR

Address: 0x4003C054 (0), 0x40064054 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–		–		–		–	
23	22	21	20	19	18	17	16
GAIN11		GAIN10		GAIN9		GAIN8	
15	14	13	12	11	10	9	8
GAIN7		GAIN6		GAIN5		GAIN4	
7	6	5	4	3	2	1	0
GAIN3		GAIN2		GAIN1		GAIN0	

This register can only be written if the WPEN bit is cleared in the AFEC Write Protection Mode Register.

- **GAINx: Gain for Channel x**

Gain applied on input of Analog Front-End.

GAINx	Gain Applied	
	DIFFx = 0(1)	DIFFx = 1(1)
0	1	1
1	2	2
2	4	4
3	4	4

Note: 1. See Section 50.7.17 "AFEC Channel Differential Register" for the description of DIFFx.

Downloaded from Arrow.com.

50.7.17 AFEC Channel Differential Register

Name: AFEC_DIFFR

Address: 0x4003C060 (0), 0x40064060 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	DIFF11	DIFF10	DIFF9	DIFF8
7	6	5	4	3	2	1	0
DIFF7	DIFF6	DIFF5	DIFF4	DIFF3	DIFF2	DIFF1	DIFF0

This register can only be written if the WPEN bit is cleared in the AFEC Write Protection Mode Register.

- **DIFFx: Differential inputs for channel x**

0: Single-ended mode.

1: Fully-differential mode.

Downloaded from Arrow.com.

50.7.18 AFEC Channel Selection Register

Name: AFEC_CSELR

Address: 0x4003C064 (0), 0x40064064 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–		CSEL			

- **CSEL: Channel Selection**

0–11: Selects the channel to be displayed in AFEC_CDR and AFEC_CAOR. To be filled with the appropriate channel number.

Downloaded from Arrow.com.

50.7.19 AFEC Channel Data Register

Name: AFEC_CDR

Address: 0x4003C068 (0), 0x40064068 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
DATA							
7	6	5	4	3	2	1	0
DATA							

- **DATA: Converted Data**

Returns the AFE conversion data corresponding to channel CSEL (configured in the AFEC Channel Selection Register).

At the end of a conversion, the converted data is loaded into one of the 12 internal registers (one for each channel) and remains in this internal register until a new conversion is completed on the same channel index. The AFEC_CDR together with AFEC_CSELR allows to multiplex all the internal channel data registers.

The data carried on AFEC_CDR is valid only if AFEC_CHSR.CH*x* bit is set (where *x* = AFEC_CSELR.CSEL field value).

Downloaded from Arrow.com.

50.7.20 AFEC Channel Offset Compensation Register

Name: AFEC_COCR

Address: 0x4003C06C (0), 0x4006406C (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	AOFF	
7	6	5	4	3	2	1	0
AOFF							

This register can only be written if the WPEN bit is cleared in the AFEC Write Protection Mode Register.

- **AOFF: Analog Offset**

Defines the analog offset to be used for channel CSEL (configured in the AFEC Channel Selection Register). This value is used as an input value for the DAC included in the AFE.

Downloaded from Arrow.com.

50.7.21 AFEC Temperature Sensor Mode Register

Name: AFEC_TEMPMR

Address: 0x4003C070 (0), 0x40064070 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	TEMPCMPMOD		–	–	–	RTCT

This register can only be written if the WPEN bit is cleared in the AFEC Write Protection Mode Register.

- **RTCT: Temperature Sensor RTC Trigger Mode**

0: The temperature sensor measure is not triggered by RTC event.

1: The temperature sensor measure is triggered by RTC event (if TRGEN = 1).

- **TEMPCMPMOD: Temperature Comparison Mode**

Value	Name	Description
0	LOW	Generates an event when the converted data is lower than the low threshold of the window.
1	HIGH	Generates an event when the converted data is higher than the high threshold of the window.
2	IN	Generates an event when the converted data is in the comparison window.
3	OUT	Generates an event when the converted data is out of the comparison window.

Downloaded from Arrow.com.

50.7.22 AFEC Temperature Compare Window Register

Name: AFEC_TEMPCWR

Address: 0x4003C074 (0), 0x40064074 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
THIGHTHRES							
23	22	21	20	19	18	17	16
THIGHTHRES							
15	14	13	12	11	10	9	8
TLOWTHRES							
7	6	5	4	3	2	1	0
TLOWTHRES							

This register can only be written if the WPEN bit is cleared in the AFEC Write Protection Mode Register.

- **TLOWTHRES: Temperature Low Threshold**

Low threshold associated to compare settings of the AFEC_TEMPMR. For comparisons less than 16 bits and signed, the sign should be extended up to the bit 15.

- **THIGHTHRES: Temperature High Threshold**

High threshold associated to compare settings of the AFEC_TEMPMR. For comparisons less than 16 bits and signed, the sign should be extended up to the bit 15.

Downloaded from Arrow.com.

50.7.23 AFEC Analog Control Register

Name: AFEC_ACR

Address: 0x4003C094 (0), 0x40064094 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	IBCTL	
7	6	5	4	3	2	1	0
–	–	–	–	PGA1EN	PGA0EN	–	–

This register can only be written if the WPEN bit is cleared in the AFEC Write Protection Mode Register.

- **PGA0EN : PGA0 Enable**

0: Programmable Gain Amplifier is disabled.

1: Programmable Gain Amplifier is enabled.

- **PGA1EN : PGA1 Enable**

0: Programmable Gain Amplifier is disabled.

1: Programmable Gain Amplifier is enabled.

- **IBCTL: AFE Bias Current Control**

Adapts performance versus power consumption. (Refer the AFE Characteristics in Section 56. ”Electrical Characteristics”.)

Downloaded from Arrow.com.

50.7.24 AFEC Sample & Hold Mode Register

Name: AFEC_SHMR

Address: 0x4003C0A0 (0), 0x400640A0 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	DUAL11	DUAL10	DUAL9	DUAL8
7	6	5	4	3	2	1	0
DUAL7	DUAL6	DUAL5	DUAL4	DUAL3	DUAL2	DUAL1	DUAL0

This register can only be written if the WPEN bit is cleared in the AFEC Write Protection Mode Register.

- **DUALx: Dual Sample & Hold for channel x**

0: Single Sample-and-Hold mode.

1: Dual Sample-and-Hold mode.

Downloaded from Arrow.com.

50.7.25 AFEC Correction Select Register

Name: AFEC_COSR

Address: 0x4003C0D0 (0), 0x400640D0 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	CSEL

This register can only be written if the WPEN bit is cleared in the AFEC Write Protection Mode Register.

- **CSEL: Sample & Hold unit Correction Select**

Selects the Sample & Hold unit to be displayed in the AFEC_CVR.

Downloaded from Arrow.com.

50.7.26 AFEC Correction Values Register

Name: AFEC_CVR

Address: 0x4003C0D4 (0), 0x400640D4 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
GAINCORR							

23	22	21	20	19	18	17	16
GAINCORR							

15	14	13	12	11	10	9	8
OFFSETCORR							

7	6	5	4	3	2	1	0
OFFSETCORR							

This register can only be written if the WPEN bit is cleared in the AFEC Write Protection Mode Register.

- **OFFSETCORR: Offset Correction**

Offset correction to apply on converted data. The offset is signed (2’s complement), only bits 0 to 11 are relevant (other bits are ignored and read as 0).

- **GAINCORR: Gain Correction**

Gain correction to apply on converted data. Only bits 0 to 11 are relevant (other bits are ignored and read as 0).

Downloaded from Arrow.com.

50.7.27 AFEC Channel Error Correction Register

Name: AFEC_CECR

Address: 0x4003C0D8 (0), 0x400640D8 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	ECORR11	ECORR10	ECORR9	ECORR8
7	6	5	4	3	2	1	0
ECORR7	ECORR6	ECORR5	ECORR4	ECORR3	ECORR2	ECORR1	ECORR0

This register can only be written if the WPEN bit is cleared in the AFEC Write Protection Mode Register.

- **ECORRx: Error Correction Enable for channel x**

0: Automatic error correction is disabled for channel x.

1: Automatic error correction is enabled for channel x.

Downloaded from Arrow.com.

50.7.28 AFEC Write Protection Mode Register

Name: AFEC_WPMR

Address: 0x4003C0E4 (0), 0x400640E4 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							

23	22	21	20	19	18	17	16
WPKEY							

15	14	13	12	11	10	9	8
WPKEY							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x414443 (“ADC” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x414443 (“ADC” in ASCII).

See Section 50.6.18 ”Register Write Protection” for the list of registers which can be protected.

- **WPKEY: Write Protect KEY**

Value	Name	Description
0x414443	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

Downloaded from Arrow.com.

50.7.29 AFEC Write Protection Status Register

Name: AFEC_WPSR

Address: 0x4003C0E8 (0), 0x400640E8 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

- **WPVS: Write Protect Violation Status**

0: No Write Protect Violation has occurred since the last read of the AFEC_WPSR.

1: A Write Protect Violation has occurred since the last read of the AFEC_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protect Violation Source**

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

Downloaded from Arrow.com.

51. Digital-to-Analog Converter Controller (DACC)

51.1 Description

The Digital-to-Analog Converter Controller (DACC) offers up to two single-ended analog outputs or one differential analog output, making it possible for the digital-to-analog conversion to drive up to two independent analog lines.

The DACC supports 12-bit resolution.

The DACC operates in Free-running mode, Max speed mode, Trigger mode or Interpolation mode.

The DACC integrates a Bypass mode which minimizes power consumption in case of a limited sampling rate conversion.

Each channel connects with a separate DMA channel. This feature reduces both power consumption and processor intervention.

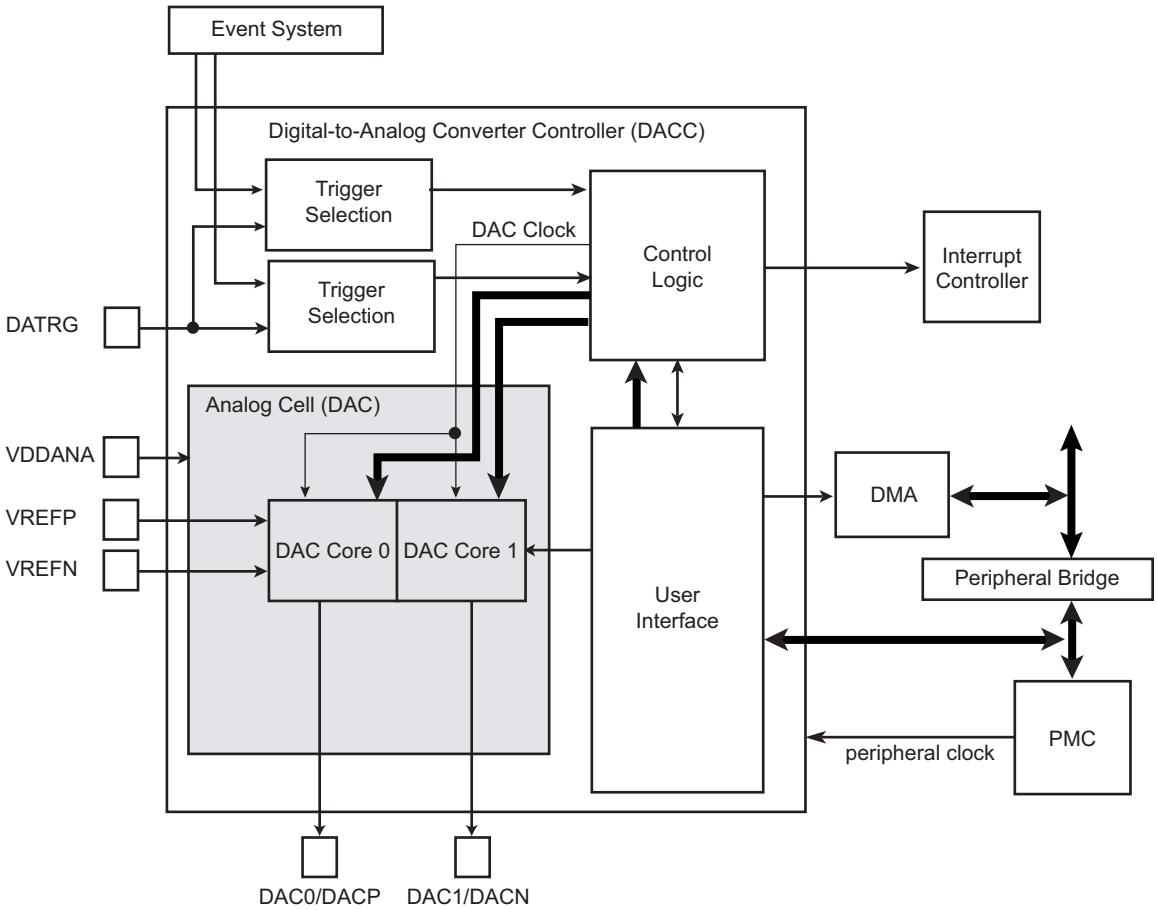
51.2 Embedded Characteristics

- Up to Two Independent Single-Ended Analog Outputs or One Differential Analog Output
- 12-bit Resolution
- Integrated Interpolation Filter with 2×, 4×, 8×, 16× or 32× Over Sampling Ratio (OSR)
- Reduced Number of System Bus Accesses (Word Transfer Mode)
- Individual Control of Each Analog Channel
- Hardware Triggers
  - One Trigger Selection Per Channel
    - External trigger pin
    - Internal events
- DMA Support
- One Internal FIFO per Channel
- Register Write Protection

Downloaded from Arrow.com.

51.3 Block Diagram

Figure 51-1. Block Diagram



51.4 Signal Description

Table 51-1. DACC Signal Description

Name	Description	Direction
DAC0/DACP	Single-ended analog output channel 0 / Positive channel of differential analog output channel	Output
DAC1/DACN	Single-ended analog output channel 1 / Negative channel of differential analog output channel	Output
DATRG	Trigger	Input
VREFP	Positive reference voltage connected to ADVREFP	Input
VREFN	Negative reference voltage connected to ADVREFN	Input

Downloaded from Arrow.com.

51.5 Product Dependencies

51.5.1 I/O Lines

The digital input DATRG is multiplexed with digital functions on the I/O line and is selected using the PIO Controller.

The analog outputs DAC0/DACP, DAC1/DACN are multiplexed with digital functions on the I/O lines .The analog outputs of the DACC drive the pads and the digital functions are not selected when the corresponding DACC channels are enabled by writing to the DACC Channel Enable Register (DACC_CHER).

Table 51-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
DACC	DAC0	PB13	X1
DACC	DAC1	PD0	X1
DACC	DATRG	PA2	C

51.5.2 Power Management

The programmer must first enable the DACC Clock in the Power Management Controller (PMC) before using the DACC.

The DACC becomes active as soon as a conversion is requested and at least one channel is enabled. The DACC is automatically deactivated when no channels are enabled.

51.5.3 Interrupt Sources

The DACC interrupt line is connected on one of the internal sources of the Interrupt controller. Using the DACC interrupt requires the Interrupt controller to be programmed first.

Table 51-3. Peripheral IDs

Instance	ID
DACC	30

51.5.4 Conversion Performances

For performance and electrical characteristics of the DACC, see the DACC Characteristics in the section “Electrical Characteristics”.

Downloaded from Arrow.com.

51.6 Functional Description

51.6.1 Digital-to-Analog Conversion

To perform conversions, the DACC_CHSR.CHx bit must be set by writing a one to DACC_CHER.CHx. If both DACC_CHSR.CHx bits are cleared, the DAC analog cell is switched off. The DACx is ready to convert once DACC_CHSR.DACRDYx is read at '1'.

The DACC divides the peripheral clock to perform conversions. This divided clock is named DAC clock. Once a conversion starts, the DACC takes 12 DAC clock periods to provide the analog result on the selected analog output.

The minimum conversion period of the DAC is exactly 12 DAC clock periods when the Max speed mode is enabled (MAXSx = 1 in the DACC Mode Register (DACC_MR)). In this case the DAC is always clocked, slightly increasing the power consumption of the DAC.

When the Max Speed mode is not used (Trigger or Free-running mode), the DAC is only clocked when a conversion is requested and a new conversion can only occur when the DAC has ended its previous conversion. The power consumption is lower but the sampling rate is lower as the controller waits for the end of conversion of the previously sent data. In this case, one conversion lasts 12 DAC clock periods plus 2 cycles of resynchronization stage.

The conversion mode of a channel can be modified only if this channel has been previously disabled.

Power consumption of the DAC can be adapted to its sampling rate via the DACC_ACR.IBCTLCHx fields.

In Bypass mode, the maximum sample rate and the power consumption of the DAC are lowered.

51.6.2 Conversion Results

When a conversion is completed, the resulting analog value is available at the selected DAC channel output. The EOC bit in the DACC Interrupt Status Register (DACC_ISR) is set.

Reading DACC_ISR clears the EOC bit.

51.6.3 Analog Output Mode Selection

The analog outputs can be set to either single-ended or differential mode with the DIFF bit in the DACC_MR.

When set to single-ended mode (DIFF = 0), each DAC channel can be configured independently.

When set to differential mode (DIFF = 1) , the analog outputs DACP and DACN are located on DAC0 and DAC1 outputs, respectively. All operations are driven by channel 0 and activating this channel automatically activates channel 1. Sending a value on channel 0 (DACP) automatically generates the complementary signal to be sent to channel 1 (DACN). The signal sent to the DAC is centered around 2048. For example, sending 3000 = 2048 + 952 to the DAC0 channel will automatically send 1096 = 2048 - 952 to the DAC1 channel.

Downloaded from Arrow.com.

51.6.4 Conversion Modes

The conversion modes available in the DACC are described below.

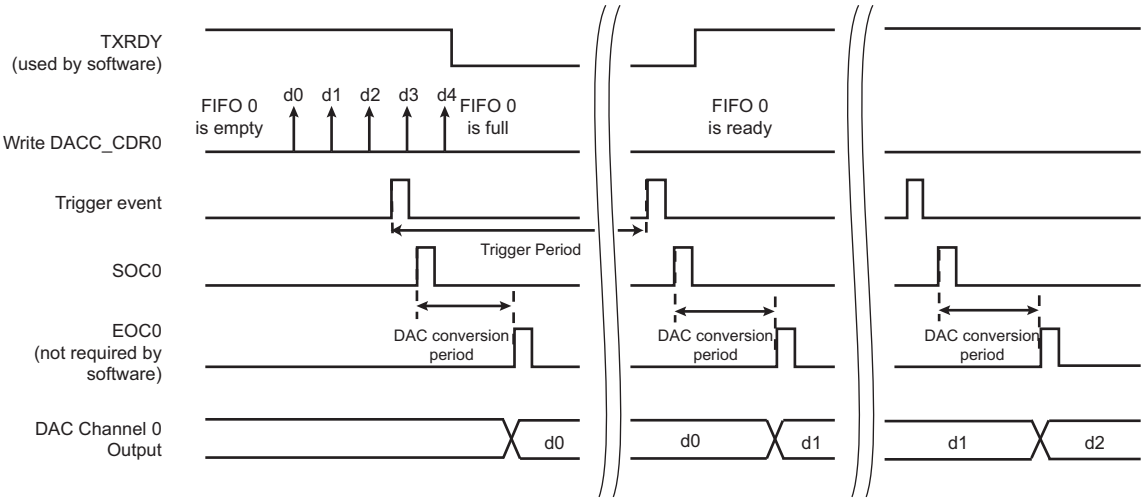
51.6.4.1 Trigger Mode

Trigger mode is enabled by setting DACC_TRIGR.TRGENx.

The conversion waits for a rising edge on the selected trigger to send the data to the DAC. In this mode, the maximum data rate (i.e., the maximum trigger event frequency) cannot exceed 12 DAC clock periods plus 2 cycles of resynchronization stage.

Note: Disabling Trigger mode (TRGENx = 0) automatically sets the DACC in Free-running or Max speed mode depending on the status of DACC_MR.MAXSx.

Figure 51-2. Conversion Sequence in Trigger Mode



Downloaded from Arrow.com.

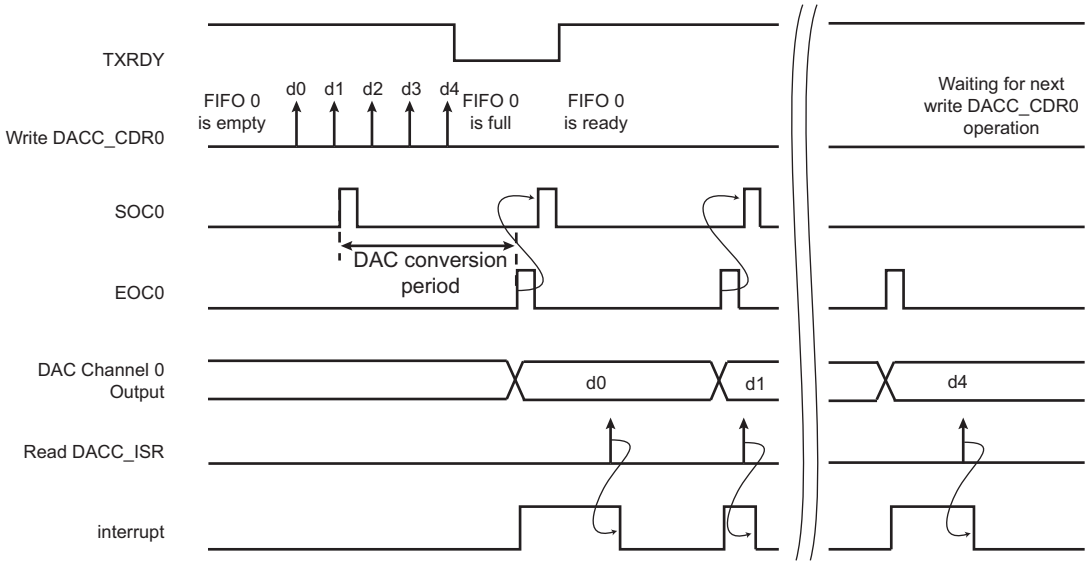
51.6.4.2 Free-Running Mode

Free-running mode is enabled by clearing DACC_TRIGR.TRGENx and DACC_MR.MAXSx.

The conversion starts as soon as at least one channel is enabled. Once data is written in the DACC Conversion Data Register (DACC_CDRx), 12 DAC clock periods later, the converted data is available at the corresponding analog output. The next data is converted only when the EOC of the previous data is set.

If the FIFO is emptied, no conversion occurs and the data is maintained at the output of the DAC.

Figure 51-3. Conversion Sequence in Free-running Mode



Downloaded from Arrow.com.

51.6.4.3 Max Speed Mode

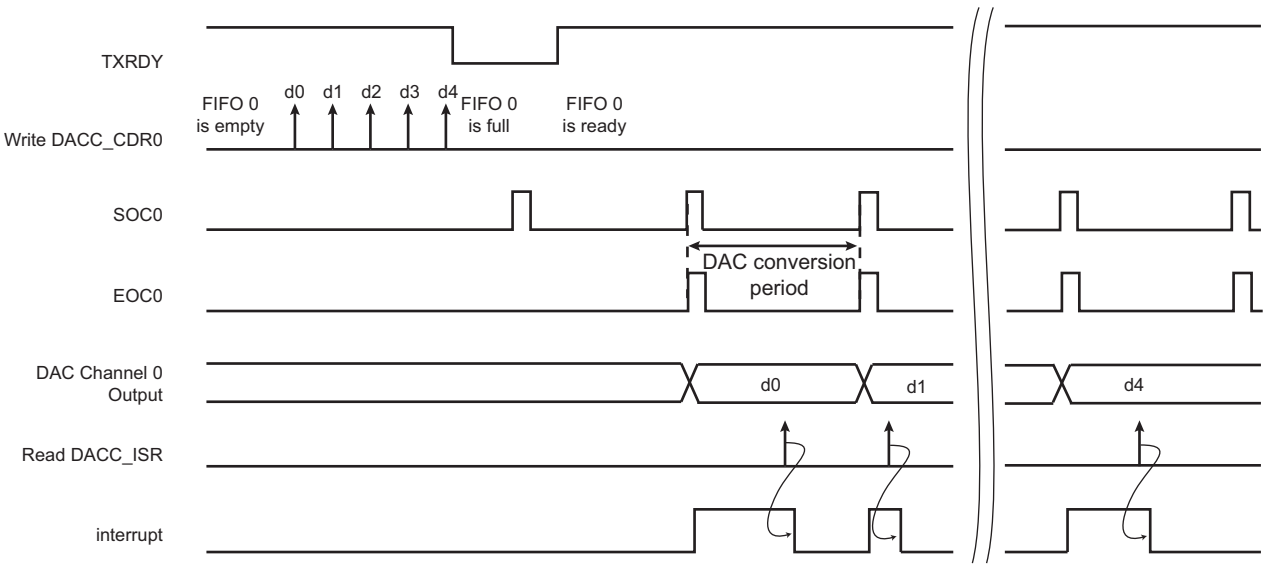
Max speed mode is enabled by setting DACC_TRIGR.TRGENx and DACC_MR.MAXSx.

The conversion rate is forced by the controller, which starts one conversion every 12 DAC clock periods. The controller does not wait for the EOC of the previous data to send a new data to the DAC and the DAC is always clocked.

If the FIFO is emptied, the controller send the last converted data to the DAC at a rate of 12 DAC clock periods.

The DACC_ACR.IBCTLCHx field must be configured for 1 MSps (see the section "Electrical Characteristics").

Figure 51-4. Conversion Sequence in Max Speed Mode



Downloaded from Arrow.com.

51.6.4.4 Bypass Mode

Bypass mode disables the DAC output buffer and thus minimizes power consumption. This mode can be used to generate slow varying signals. Refer to the DAC Characteristics in Section 56. "Electrical Characteristics".

To enter this mode, Free-running mode must be selected and the DACC_ACR.IBCTLCHx field configured in Bypass mode.

51.6.4.5 Interpolation Mode

The DACC integrates interpolation filters that allow OSR of 2×, 4×, 8×, 16× or 32×. This mode can be used only if Trigger mode is enabled and value in the field OSRx is not '0'. The OSR of the interpolator is configured in the OSRx field in the DACC Trigger Register (DACC_TRIGR).

The data is sampled once every OSR trigger event and then recomputed at the trigger sample rate using a third-order SINC filter. This reduces the number of accesses to the DACC and increases the signal-to-noise ratio (SNR) of the converted output signal.

Figure 51-5 to Figure 51-9 show the spectral mask of the SINC filter depending on the selected OSR. f_s is the sampling frequency of the input signal which corresponds to the trigger frequency divided by OSR.

Figure 51-5. Interpolator Spectral Mask for OSR = 2

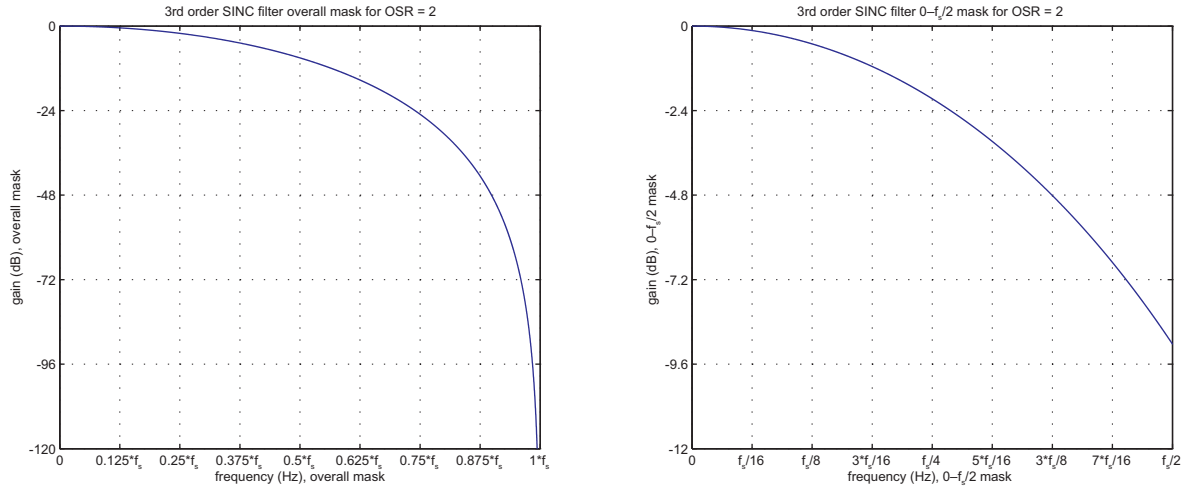
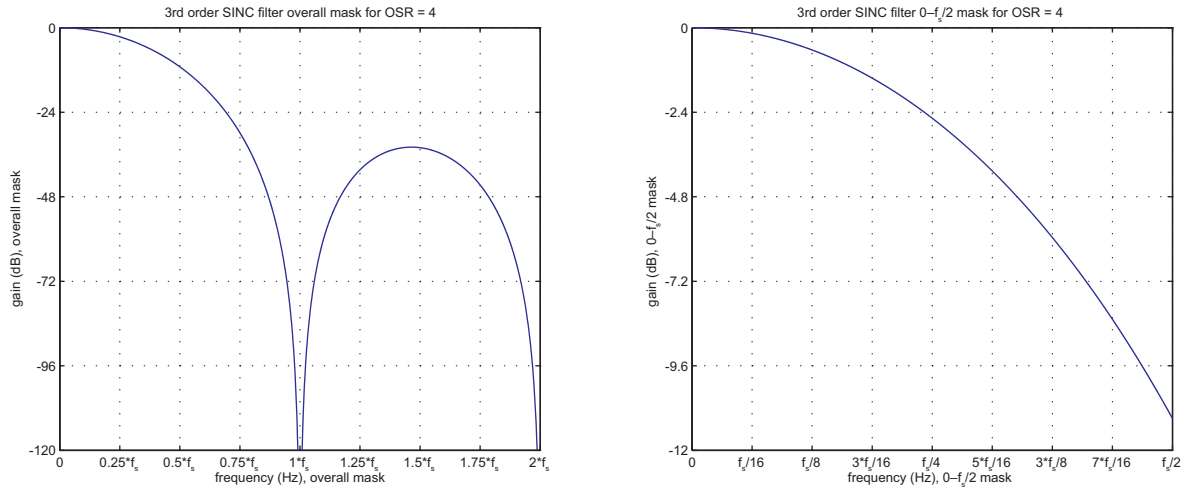


Figure 51-6. Interpolator Spectral Mask for OSR = 4



Downloaded from Arrow.com.

Figure 51-7. Interpolator Spectral Mask for OSR = 8

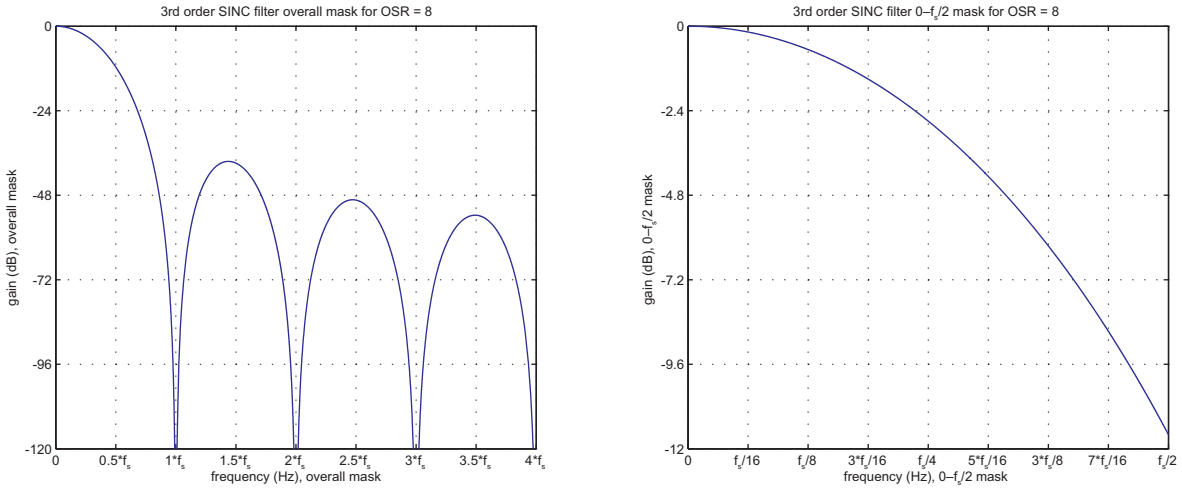


Figure 51-8. Interpolator Spectral Mask for OSR = 16

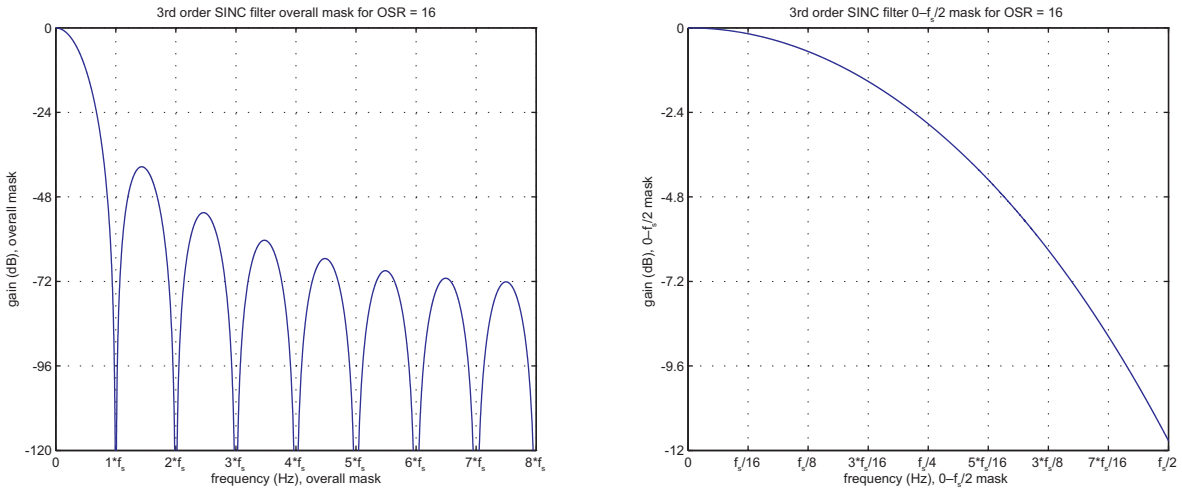
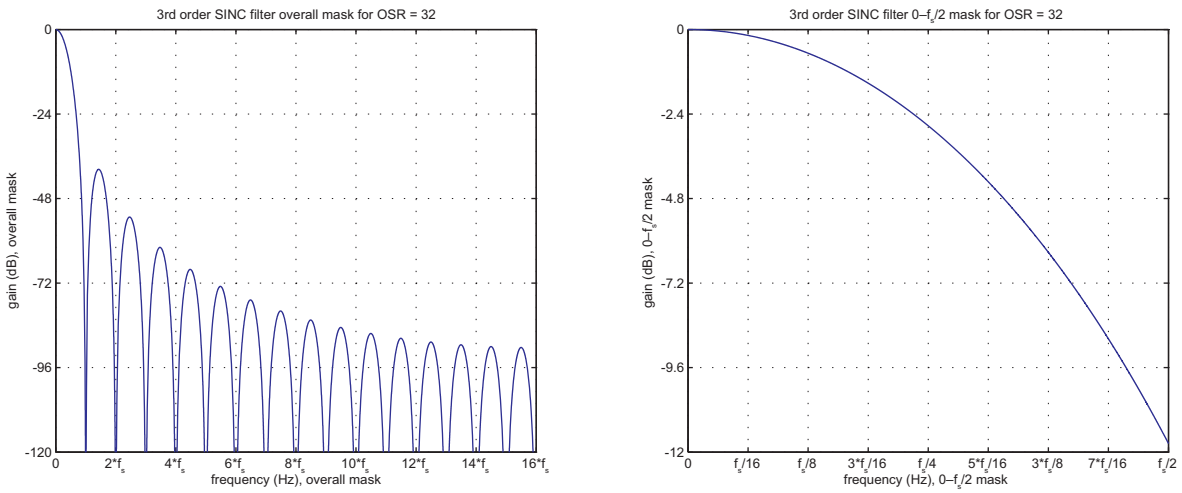


Figure 51-9. Interpolator Spectral Mask for OSR = 32



Downloaded from Arrow.com.

51.6.5 Conversion FIFO

Each channel embeds a four half-word FIFO to handle the data to be converted.

When the TXRDY flag of a channel in the DACC_ISR is active, the DACC is ready to accept conversion requests by writing data into the corresponding DACC_CDRx. Data which cannot be converted immediately are stored in the FIFO of the corresponding channel.

When the FIFO is full or the DACC is not ready to accept conversion requests, the TXRDY flag is inactive.

The DACC also offers the possibility of writing two data words in one access by setting the bit WORD in the DACC_MR. In this case, bits 11:0 contain the first data to be converted and bits 27:16 contain the second data to be converted. The two data are written into the FIFO of the selected channel. The TXRDY flag takes into account this double write access. Changing this access mode implies first switching off all channels.

WARNING: Writing in DACC_CDRx while TXRDY flag is inactive will corrupt FIFO data.

51.6.6 Register Write Protection

To prevent any single software error from corrupting DACC behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the DACC Write Protection Mode Register (DACC_WPMR).

If a write access to a write-protected register is detected, the WPVS bit in the DACC Write Protection Status Register (DACC_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading the DACC_WPSR.

The following registers can be write-protected :

- DACC Mode Register
- DACC Channel Enable Register
- DACC Channel Disable Register
- DACC Analog Current Register
- DACC Trigger Register

Downloaded from Arrow.com.

51.7 Digital-to-Analog Converter Controller (DACC) User Interface

Table 51-4. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	DACC_CR	Write-only	–
0x04	Mode Register	DACC_MR	Read/Write	0x00000000
0x08	Trigger Register	DACC_TRIGR	Read/Write	0x00000000
0x0C	Reserved	–	–	–
0x10	Channel Enable Register	DACC_CHER	Write-only	–
0x14	Channel Disable Register	DACC_CHDR	Write-only	–
0x18	Channel Status Register	DACC_CHSR	Read-only	0x00000000
0x1C	Conversion Data Register 0	DACC_CDR0	Write-only	–
0x20	Conversion Data Register 1	DACC_CDR1	Write-only	–
0x24	Interrupt Enable Register	DACC_IER	Write-only	–
0x28	Interrupt Disable Register	DACC_IDR	Write-only	–
0x2C	Interrupt Mask Register	DACC_IMR	Read-only	0x00000000
0x30	Interrupt Status Register	DACC_ISR	Read-only	0x00000000
0x34–0x90	Reserved	–	–	–
0x94	Analog Current Register	DACC_ACR	Read/Write	0x00000000
0x98–0xE0	Reserved	–	–	–
0xE4	Write Protection Mode Register	DACC_WPMR	Read/Write	0x00000000
0xE8	Write Protection Status Register	DACC_WPSR	Read-only	0x00000000
0xEC–0xFC	Reserved	–	–	–

Downloaded from Arrow.com.

51.7.1 DACC Control Register

Name: DACC_CR

Address: 0x40040000

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	SWRST

- **SWRST: Software Reset**

0: No effect.

1: Resets the DACC simulating a hardware reset.

Downloaded from Arrow.com.

51.7.2 DACC Mode Register

Name: DACC_MR

Address: 0x40040004

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	PRESCALER			
23	22	21	20	19	18	17	16
DIFF	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	ZERO	WORD	–	–	MAXS1	MAXS0

This register can only be written if the WPEN bit is cleared in the DACC Write Protection Mode Register.

- **MAXSx: Max Speed Mode for Channel x**

Value	Name	Description
0	TRIG_EVENT	Trigger mode or Free-running mode enabled. (See TRGENx.DACC_TRIGR.)
1	MAXIMUM	Max speed mode enabled.

- **WORD: Word Transfer Mode**

Value	Name	Description
0	DISABLED	One data to convert is written to the FIFO per access to DACC.
1	ENABLED	Two data to convert are written to the FIFO per access to DACC (reduces the number of requests to DMA and the number of system bus accesses).

- **ZERO: Must always be written to 0.**

- **DIFF: Differential Mode**

Value	Name	Description
0	DISABLED	DAC0 and DAC1 are single-ended outputs.
1	ENABLED	DACP and DACN are differential outputs. The differential level is configured by the channel 0 value.

- **PRESCALER: Peripheral Clock to DAC Clock Ratio**

  This field defines the division ratio between the peripheral clock and the DAC clock respecting the following formula:

$$\text{PRESCALER} = \left(\frac{f_{\text{peripheral clock}}}{f_{DAC}}\right) - 2$$

Downloaded from Arrow.com.

51.7.3 DACC Trigger Register

Name: DACC_TRIGR

Address: 0x40040008

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	OSR1			–	OSR0		
15	14	13	12	11	10	9	8
–	–	–	–	–	TRGSEL1		
7	6	5	4	3	2	1	0
–	TRGSEL0			–	–	TRGEN1	TRGEN0

This register can only be written if the WPEN bit is cleared in the DACC Write Protection Mode Register.

- **TRGENx: Trigger Enable of Channel x**

Value	Name	Description
0	DIS	Trigger mode disabled. DACC is in Free-running mode or Max speed mode.
1	EN	Trigger mode enabled.

- **TRGSELx: Trigger Selection of Channel x**

Value	Name	Description
0	TRGSEL0	DATRG
1	TRGSEL1	TC0 output
2	TRGSEL2	TC1 output
3	TRGSEL3	TC2 output
4	TRGSEL4	PWM0 event 0
5	TRGSEL5	PWM0 event 1
6	TRGSEL6	PWM1 event 0
7	TRGSEL7	PWM1 event 1

- **OSRx: Over Sampling Ratio of Channel x**

Value	Name	Description
0	OSR_1	OSR = 1
1	OSR_2	OSR = 2
2	OSR_4	OSR = 4
3	OSR_8	OSR = 8
4	OSR_16	OSR = 16
5	OSR_32	OSR = 32

Downloaded from Arrow.com.

51.7.4 DACC Channel Enable Register

Name: DACC_CHER

Address: 0x40040010

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	CH1	CH0

This register can only be written if the WPEN bit is cleared in the DACC Write Protection Mode Register.

- **CHx: Channel x Enable**

0: No effect.

1: Enables the corresponding channel.

Downloaded from Arrow.com.

51.7.5 DACC Channel Disable Register

Name: DACC_CHDR

Address: 0x40040014

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	CH1	CH0

This register can only be written if the WPEN bit is cleared in the DACC Write Protection Mode Register.

- **CHx: Channel x Disable**

0: No effect.

1: Disables the corresponding channel.

WARNING: If the corresponding channel is disabled during a conversion or if it is disabled then re-enabled during a conversion, its associated analog value and its corresponding EOC flags in DACC_ISR are unpredictable.

Downloaded from Arrow.com.

51.7.6 DACC Channel Status Register

Name: DACC_CHSR

Address: 0x40040018

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	DACRDY1	DACRDY0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	CH1	CH0

- **CHx: Channel x Status**

0: Corresponding channel is disabled.

1: Corresponding channel is enabled.

- **DACRDYx: DAC Ready Flag**

0: The DACx is not yet ready to receive data.

1: The DACx is ready to receive data.

Downloaded from Arrow.com.

51.7.7 DACC Conversion Data Register

Name: DACC_CDRx

Address: 0x4004001C

Access: Write-only

31	30	29	28	27	26	25	24
DATA1							
23	22	21	20	19	18	17	16
DATA1							
15	14	13	12	11	10	9	8
DATA0							
7	6	5	4	3	2	1	0
DATA0							

- **DATA0: Data to Convert for channel x**

DATA0 is written to the FIFO of channel x.

- **DATA1: Data to Convert for channel x**

If DACC_MR.WORD is set, DATA1 is written to the FIFO of channel x after DATA0.

Downloaded from Arrow.com.

51.7.8 DACC Interrupt Enable Register

Name: DACC_IER

Address: 0x40040024

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	EOC1	EOC0			TXRDY1	TXRDY0

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

- **TXRDYx: Transmit Ready Interrupt Enable of channel x**
- **EOCx: End of Conversion Interrupt Enable of channel x**

Downloaded from Arrow.com.

51.7.9 DACC Interrupt Disable Register

Name: DACC_IDR

Address: 0x40040028

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	EOC1	EOC0			TXRDY1	TXRDY0

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

- **TXRDYx: Transmit Ready Interrupt Disable of channel x**
- **EOCx: End of Conversion Interrupt Disable of channel x**

Downloaded from Arrow.com.

51.7.10 DACC Interrupt Mask Register

Name: DACC_IMR

Address: 0x4004002C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	EOC1	EOC0			TXRDY1	TXRDY0

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

- **TXRDYx: Transmit Ready Interrupt Mask of channel x**
- **EOCx: End of Conversion Interrupt Mask of channel x**

Downloaded from Arrow.com.

51.7.11 DACC Interrupt Status Register

Name: DACC_ISR

Address: 0x40040030

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	EOC1	EOC0			TXRDY1	TXRDY0

- **TXRDYx: Transmit Ready Interrupt Flag of channel x**

0: DACC is not ready to accept new conversion requests.

1: DACC is ready to accept new conversion requests.

- **EOCx: End of Conversion Interrupt Flag of channel x**

0: No conversion has been performed since the last read of DACC_ISR.

1: At least one conversion has been performed since the last read of DACC_ISR.

Downloaded from Arrow.com.

51.7.12 DACC Analog Current Register

Name: DACC_ACR

Address: 0x40040094

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	IBCTLCH1		IBCTLCH0	

This register can only be written if the WPEN bit is cleared in the DACC Write Protection Mode Register.

- **IBCTLCHx: Analog Output Current Control**

Allows to adapt the slew rate of the analog output. For more details, refer to the Section 56. "Electrical Characteristics".

Downloaded from Arrow.com.

51.7.13 DACC Write Protection Mode Register

Name: DACC_WPMR

Address: 0x400400E4

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							
23	22	21	20	19	18	17	16
WPKEY							
15	14	13	12	11	10	9	8
WPKEY							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x444143 (“DAC” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x444143 (“DAC” in ASCII).

See Section 51.6.6 “Register Write Protection” for list of write-protected registers.

- **WPKEY: Write Protect Key**

Value	Name	Description
0x444143	PASSWD	Writing any other value in this field aborts the write operation of bit WPEN. Always reads as 0.

Downloaded from Arrow.com.

51.7.14 DACC Write Protection Status Register

Name: DACC_WPSR

Address: 0x400400E8

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
WPVSRC							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

- **WPVS: Write Protection Violation Status**

0: No write protection violation has occurred since the last read of the DACC_WPSR.

1: A write protection violation has occurred since the last read of the DACC_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protection Violation Source**

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

Downloaded from Arrow.com.

52. Analog Comparator Controller (ACC)

52.1 Description

The Analog Comparator Controller (ACC) configures the analog comparator and generates an interrupt depending on user settings. The analog comparator embeds two 8-to-1 multiplexers that generate two internal inputs. These inputs are compared, resulting in a compare output. The hysteresis level, edge detection and polarity are configurable.

The ACC also generates a compare event which can be used by the Pulse Width Modulator (PWM).

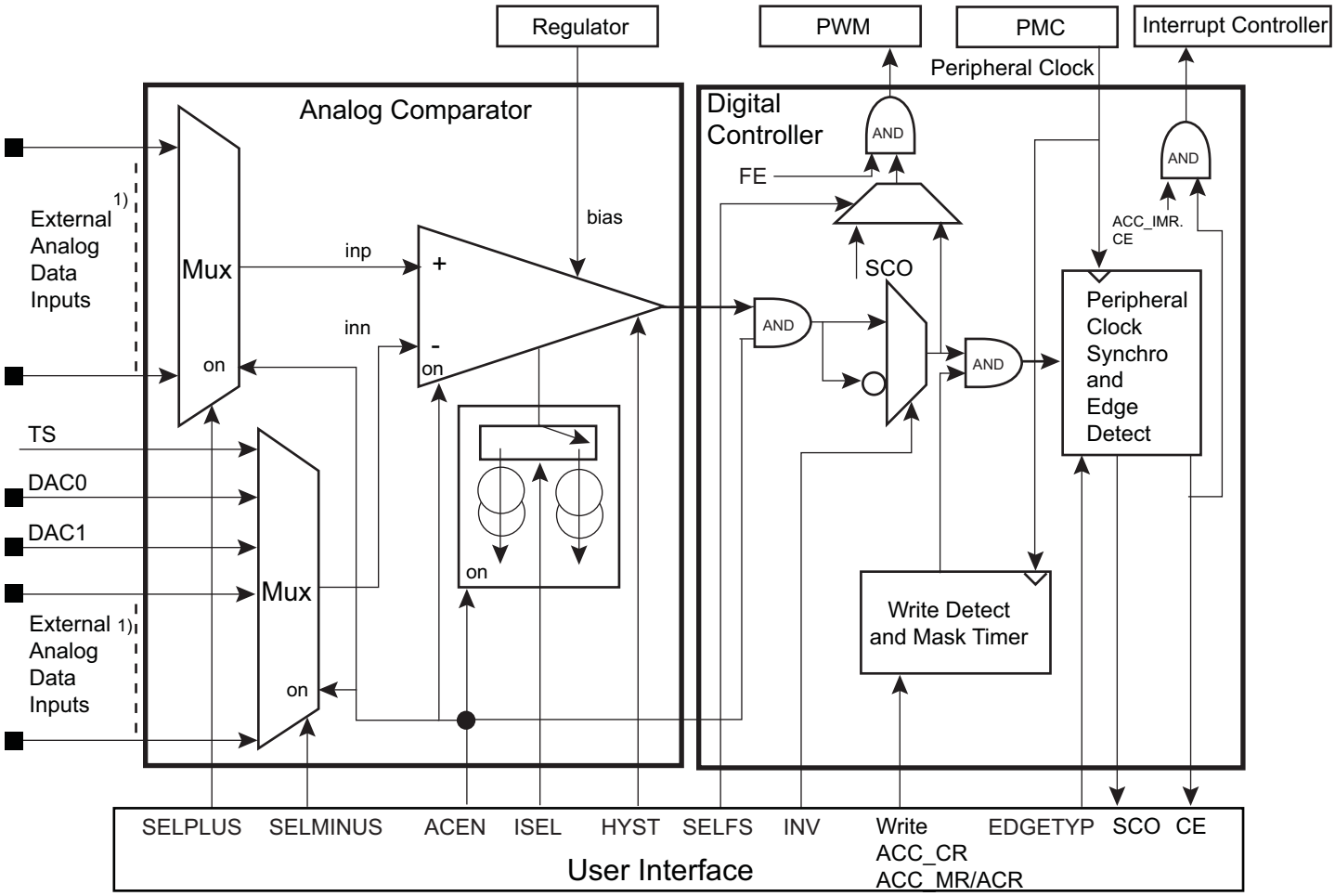
52.2 Embedded Characteristics

- Eight User Analog Inputs Selectable for Comparison
- Four Voltage References Selectable for Comparison: Temperature Sensor (TS), External Voltage Reference, DAC0 and DAC1
- Interrupt Generation
- Compare Event Fault Generation for PWM

Downloaded from Arrow.com.

52.3 Block Diagram

Figure 52-1. Analog Comparator Controller Block Diagram



52.4 Signal Description

Table 52-1. ACC Signal Description

Pin Name	Description	Type
AFE0_AD[5:0]	External analog data inputs	Input
AFE1_AD[1:0]		
TS	On-chip temperature sensor	Input
VREFP	AFE and DAC voltage reference	Input
DAC0, DAC1	On-chip DAC inputs	Input

Downloaded from Arrow.com.

52.5 Product Dependencies

52.5.1 I/O Lines

The analog input pins (AFE0_AD[5:0], AFE1_AD[1:0] and DAC0–1) are multiplexed with digital functions (PIO) on the IO line. By writing the SELMINUS and SELPLUS fields in the ACC Mode Register (ACC_MR), the associated IO lines are set to Analog mode.

52.5.2 Power Management

The ACC is clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the ACC clock.

Note that the voltage regulator must be activated to use the analog comparator.

52.5.3 Interrupt

The ACC has an interrupt line connected to the Interrupt Controller (IC). In order to handle interrupts, the Interrupt Controller must be programmed before configuring the ACC.

Table 52-2. Peripheral IDs

Instance	ID
ACC	33

52.5.4 Fault Output

The ACC has the FAULT output connected to the FAULT input of PWM. Refer to chapter Section 52.6.4 “Fault Mode” and the implementation of the PWM in the product.

Downloaded from Arrow.com.

52.6 Functional Description

52.6.1 Description

The Analog Comparator Controller (ACC) controls the analog comparator settings and performs post-processing of the analog comparator output.

When the analog comparator settings are modified, the output of the analog cell may be invalid. The ACC masks the output for the invalid period.

A comparison flag is triggered by an event on the output of the analog comparator and an interrupt is generated. The event on the analog comparator output can be selected among falling edge, rising edge or any edge.

The ACC registers are listed in Table 52-3.

52.6.2 Analog Settings

The user can select the input hysteresis and configure two different options, characterized as follows:

- High-speed: shortest propagation delay/highest current consumption
- Low-power: longest propagation delay/lowest current consumption

52.6.3 Output Masking Period

As soon as the analog comparator settings change, the output is invalid for a duration depending on ISEL current.

A masking period is automatically triggered as soon as a write access is performed on the ACC_MR or ACC Analog Control Register (ACC_ACR) (whatever the register data content).

When ISEL = 0, the mask period is $8 \times t_{\text{peripheral clock}}$.

When ISEL = 1, the mask period is $128 \times t_{\text{peripheral clock}}$.

The masking period is reported by reading a negative value (bit 31 set) on the ACC Interrupt Status Register (ACC_ISR).

52.6.4 Fault Mode

In Fault mode, a comparison match event is communicated by the ACC fault output which is directly and internally connected to a PWM fault input.

The source of the fault output can be configured as either a combinational value derived from the analog comparator output or as the peripheral clock resynchronized value (Refer to Figure 52-1 “Analog Comparator Controller Block Diagram”).

52.6.5 Register Write Protection

To prevent any single software error from corrupting ACC behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the ACC Write Protection Mode Register (ACC_WPMR).

If a write access to a write-protected register is detected, the WPVS flag in the ACC Write Protection Status Register (ACC_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading the ACC_WPSR.

The following registers can be write-protected:

- ACC Mode Register
- ACC Analog Control Register

Downloaded from Arrow.com.

52.7 Analog Comparator Controller (ACC) User Interface

Table 52-3. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	ACC_CR	Write-only	–
0x04	Mode Register	ACC_MR	Read/Write	0
0x08–0x20	Reserved	–	–	–
0x24	Interrupt Enable Register	ACC_IER	Write-only	–
0x28	Interrupt Disable Register	ACC_IDR	Write-only	–
0x2C	Interrupt Mask Register	ACC_IMR	Read-only	0
0x30	Interrupt Status Register	ACC_ISR	Read-only	0
0x34–0x90	Reserved	–	–	–
0x94	Analog Control Register	ACC_ACR	Read/Write	0
0x98–0xE0	Reserved	–	–	–
0xE4	Write Protection Mode Register	ACC_WPMR	Read/Write	0
0xE8	Write Protection Status Register	ACC_WPSR	Read-only	0
0xEC–0xFC	Reserved	–	–	–

Downloaded from Arrow.com.

52.7.1 ACC Control Register

Name: ACC_CR

Address: 0x40044000

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	SWRST

- **SWRST: Software Reset**

0: No effect.

1: Resets the module.

Downloaded from Arrow.com.

52.7.2 ACC Mode Register

Name: ACC_MR

Address: 0x40044004

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	FE	SELFS	INV	–	EDGETYP		ACEN
7	6	5	4	3	2	1	0
–	SELPLUS			–	SELMINUS		

This register can only be written if the WPEN bit is cleared in the ACC Write Protection Mode Register.

- **SELMINUS: Selection for Minus Comparator Input**

0..7: Selects the input to apply on analog comparator SELMINUS comparison input.

Value	Name	Description
0	TS	Select TS
1	VREFP	Select VREFP
2	DAC0	Select DAC0
3	DAC1	Select DAC1
4	AFE0_AD0	Select AFE0_AD0
5	AFE0_AD1	Select AFE0_AD1
6	AFE0_AD2	Select AFE0_AD2
7	AFE0_AD3	Select AFE0_AD3

- **SELPLUS: Selection For Plus Comparator Input**

0..7: Selects the input to apply on analog comparator SELPLUS comparison input.

Value	Name	Description
0	AFE0_AD0	Select AFE0_AD0
1	AFE0_AD1	Select AFE0_AD1
2	AFE0_AD2	Select AFE0_AD2
3	AFE0_AD3	Select AFE0_AD3
4	AFE0_AD4	Select AFE0_AD4
5	AFE0_AD5	Select AFE0_AD5
6	AFE1_AD0	Select AFE1_AD0
7	AFE1_AD1	Select AFE1_AD1

- **ACEN: Analog Comparator Enable**

0 (DIS): Analog comparator disabled.

1 (EN): Analog comparator enabled.

Downloaded from Arrow.com.

- **EDGETYP: Edge Type**

Value	Name	Description
0	RISING	Only rising edge of comparator output
1	FALLING	Falling edge of comparator output
2	ANY	Any edge of comparator output

- **INV: Invert Comparator Output**

0 (DIS): Analog comparator output is directly processed.

1 (EN): Analog comparator output is inverted prior to being processed.

- **SELFS: Selection Of Fault Source**

0 (CE): The CE flag is used to drive the FAULT output.

1 (OUTPUT): The output of the analog comparator flag is used to drive the FAULT output.

- **FE: Fault Enable**

0 (DIS): The FAULT output is tied to 0.

1 (EN): The FAULT output is driven by the signal defined by SELFS.

Downloaded from Arrow.com.

52.7.3 ACC Interrupt Enable Register

Name: ACC_IER

Address: 0x40044024

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	CE

- **CE: Comparison Edge**

0: No effect.

1: Enables the interrupt when the selected edge (defined by EDGETYP) occurs.

Downloaded from Arrow.com.

52.7.4 ACC Interrupt Disable Register

Name: ACC_IDR

Address: 0x40044028

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	CE

- **CE: Comparison Edge**

0: No effect.

1: Disables the interrupt when the selected edge (defined by EDGETYP) occurs.

52.7.5 ACC Interrupt Mask Register

Name: ACC_IMR

Address: 0x4004402C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	CE

- **CE: Comparison Edge**

0: The interrupt is disabled.

1: The interrupt is enabled.

Downloaded from Arrow.com.

52.7.6 ACC Interrupt Status Register

Name: ACC_ISR

Address: 0x40044030

Access: Read-only

31	30	29	28	27	26	25	24
MASK	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	SCO	CE

- **CE: Comparison Edge (cleared on read)**

0: No edge occurred (defined by EDGETYP) on analog comparator output since the last read of ACC_ISR.

1: A selected edge (defined by EDGETYP) on analog comparator output occurred since the last read of ACC_ISR.

- **SCO: Synchronized Comparator Output**

Returns an image of the analog comparator output after being pre-processed (refer to Figure 52-1 on page 1620).

If INV = 0

SCO = 0 if inn > inp

SCO = 1 if inp > inn

If INV = 1

SCO = 1 if inn > inp

SCO = 0 if inp > inn

- **MASK: Flag Mask**

0: The CE flag and SCO value are valid.

1: The CE flag and SCO value are invalid.

Downloaded from Arrow.com.

52.7.7 ACC Analog Control Register

Name: ACC_ACR

Address: 0x40044094

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	HYST		ISEL

This register can only be written if the WPEN bit is cleared in ACC Write Protection Mode Register.

- **ISEL: Current Selection**

Refer to the section on ACC characteristics in Section 56. “Electrical Characteristics” of this datasheet.

0 (LOPW): Low-power option.

1 (HISP): High-speed option.

- **HYST: Hysteresis Selection**

0 to 3: Refer to the section on ACC characteristics in Section 56. “Electrical Characteristics” of this datasheet.

Downloaded from Arrow.com.

52.7.8 ACC Write Protection Mode Register

Name: ACC_WPMR

Address: 0x400440E4

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							
23	22	21	20	19	18	17	16
WPKEY							
15	14	13	12	11	10	9	8
WPKEY							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x414343 (“ACC” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x414343 (“ACC” in ASCII).

See “Register Write Protection” on page 1622 for the list of registers that can be write-protected.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x414343	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

Downloaded from Arrow.com.

52.7.9 ACC Write Protection Status Register

Name: ACC_WPSR

Address: 0x400440E8

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

- **WPVS: Write Protection Violation Status**

0: No write protection violation has occurred since the last read of ACC_WPSR.

1: A write protection violation (WPEN = 1) has occurred since the last read of ACC_WPSR.

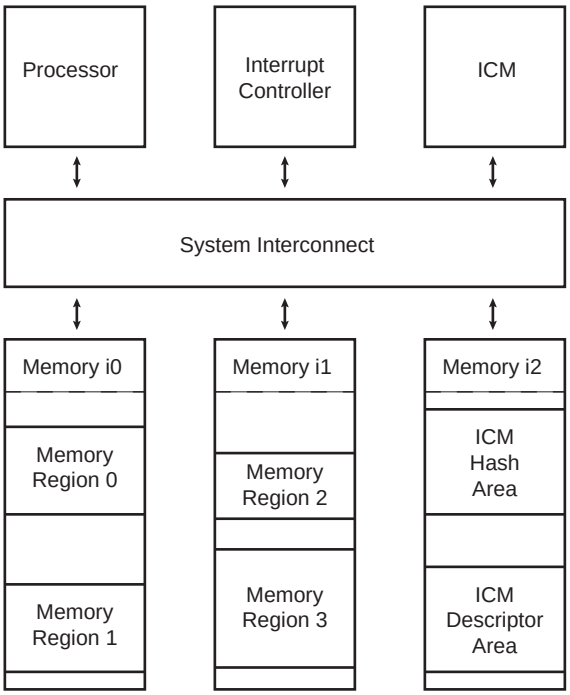
Downloaded from Arrow.com.

53. Integrity Check Monitor (ICM)

53.1 Description

The Integrity Check Monitor (ICM) is a DMA controller that performs hash calculation over multiple memory regions through the use of transfer descriptors located in memory (ICM Descriptor Area). The Hash function is based on the Secure Hash Algorithm (SHA). The ICM controller integrates two modes of operation. The first one is used to hash a list of memory regions and save the digests to memory (ICM Hash Area). The second operation mode is an active monitoring of the memory. In that mode, the hash function is evaluated and compared to the digest located at a predefined memory address (ICM Hash Area). If a mismatch occurs, an interrupt is raised. See Figure 53-1 for an example of four-region monitoring. Hash and Descriptor areas are located in Memory instance i2, and the four regions are split in memory instances i0 and i1.

Figure 53-1. Four-region Monitoring Example



The ICM SHA engine is compliant with the American *FIPS (Federal Information Processing Standard) Publication 180-2* specification.

The following terms are concise definitions of the ICM concepts used throughout this document:

- Region—a partition of instruction or data memory space
- Region Descriptor—a data structure stored in memory, defining region attributes
- Region Attributes—region start address, region size, region SHA engine processing mode, Write Back or Compare function mode
- Context Registers—a set of ICM non-memory-mapped, internal registers which are automatically loaded, containing the attributes of the region being processed
- Main List—a list of region descriptors. Each element associates the start address of a region with a set of attributes.
- Secondary List—a linked list defined on a per region basis that describes the memory layout of the region (when the region is non-contiguous)
- Hash Area—predefined memory space where the region hash results (digest) are stored

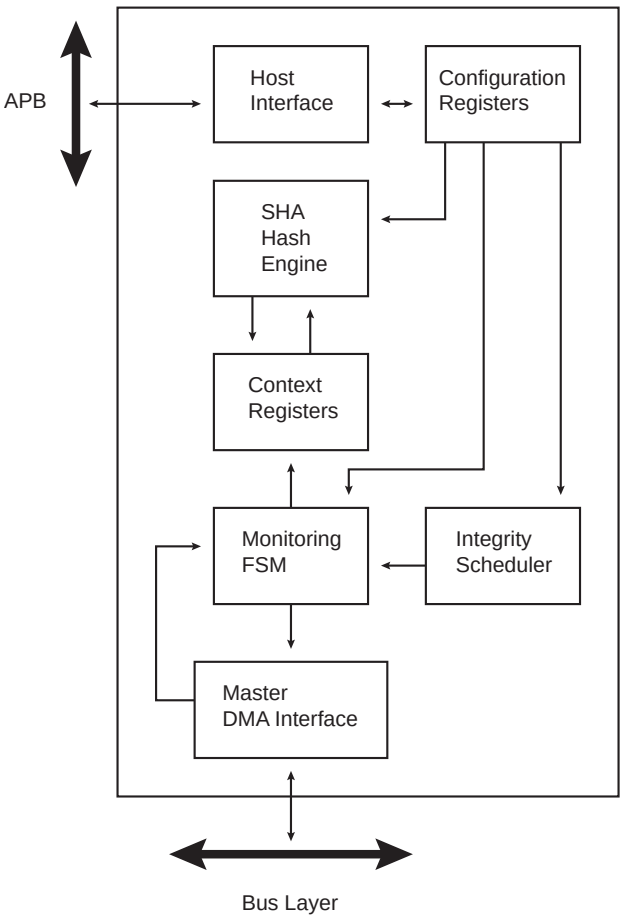
Downloaded from Arrow.com.

53.2 Embedded Characteristics

- DMA AHB master interface
- Supports monitoring of up to 4 Non-Contiguous Memory Regions
- Supports block gathering through the use of linked list
- Supports Secure Hash Algorithm (SHA1, SHA224, SHA256)
- Compliant with *FIPS Publication 180-2*
- Configurable Processing Period:
  - When SHA1 algorithm is processed, the runtime period is either 85 or 209 clock cycles.
  - When SHA256 or SHA224 algorithm is processed, the runtime period is either 72 or 194 clock cycles.
- Programmable Bus burden

53.3 Block Diagram

Figure 53-2. Integrity Check Monitor Block Diagram



Downloaded from Arrow.com.

53.4 Product Dependencies

53.4.1 Power Management

The peripheral clock is not continuously provided to the ICM. The programmer must first enable the ICM clock in the Power Management Controller (PMC) before using the ICM.

53.4.2 Interrupt Sources

The ICM interface has an interrupt line connected to the Interrupt Controller.

Handling the ICM interrupt requires programming the interrupt controller before configuring the ICM.

Table 53-1. Peripheral IDs

Instance	ID
ICM	32

Downloaded from Arrow.com.

53.5 Functional Description

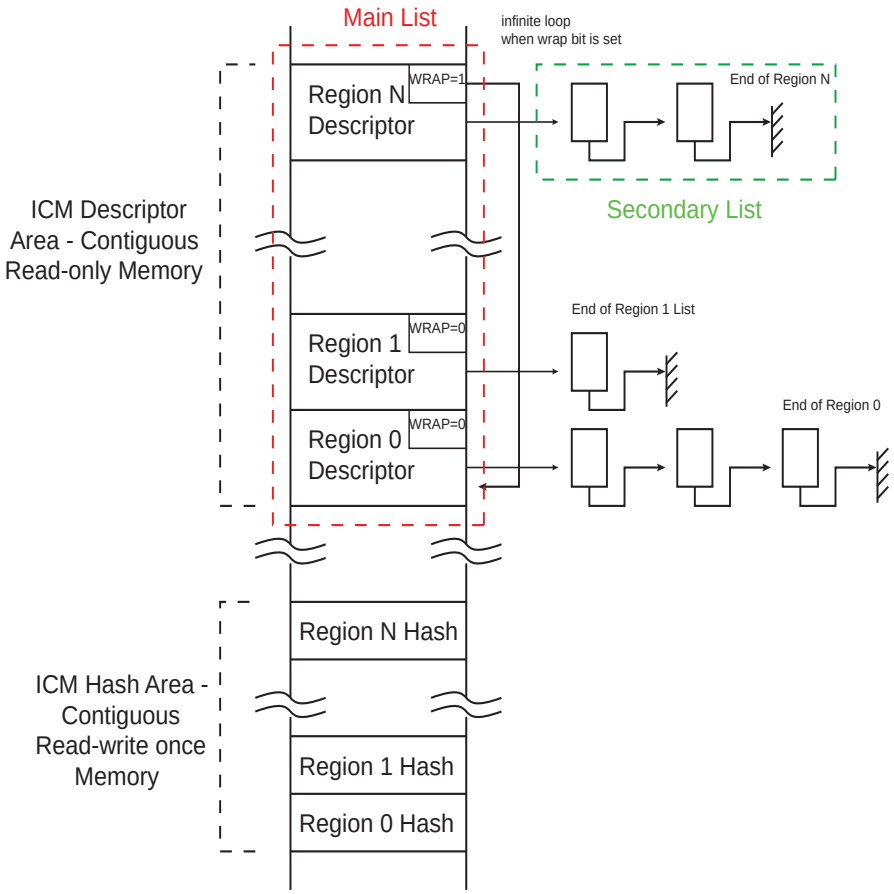
53.5.1 Overview

The Integrity Check Monitor (ICM) is a DMA controller that performs SHA-based memory hashing over memory regions. As shown in Figure 53-2, it integrates a DMA interface, a Monitoring Finite State Machine (FSM), an integrity scheduler, a set of context registers, a SHA engine, an interface for configuration and status registers.

The ICM integrates a Secure Hash Algorithm Engine (SHA). This engine requires a message padded according to FIPS180-2 specification when used as a SHA calculation unit only. Otherwise, if the ICM is used as integrated check for memory content, the padding is not mandatory. The SHA module produces an N-bit message digest each time a block is read and a processing period ends. N is 160 for SHA1, 224 for SHA224, 256 for SHA256.

When the ICM module is enabled, it sequentially retrieves a circular list of region descriptors from the memory (Main List described in Figure 53-3). Up to four regions may be monitored. Each region descriptor is composed of four words indicating the layout of the memory region (see Figure 53-4). It also contains the hashing engine configuration on a per region basis. As soon as the descriptor is loaded from the memory and context registers are updated with the data structure, the hashing operation starts. A programmable number of blocks (see TRSIZE field of the ICM_RCTRL structure member) is transferred from the memory to the SHA engine. When the desired number of blocks have been transferred, the digest is whether moved to memory (Write Back function) or compared with a digest reference located in the system memory (Compare function). If a digest mismatch occurs, an interrupt is triggered if unmasked. The ICM module passes through the region descriptor list until the end of the list marked by an End of List bit set to one. To continuously monitor the list of regions, the WRAP bit must be set to one in the last data structure.

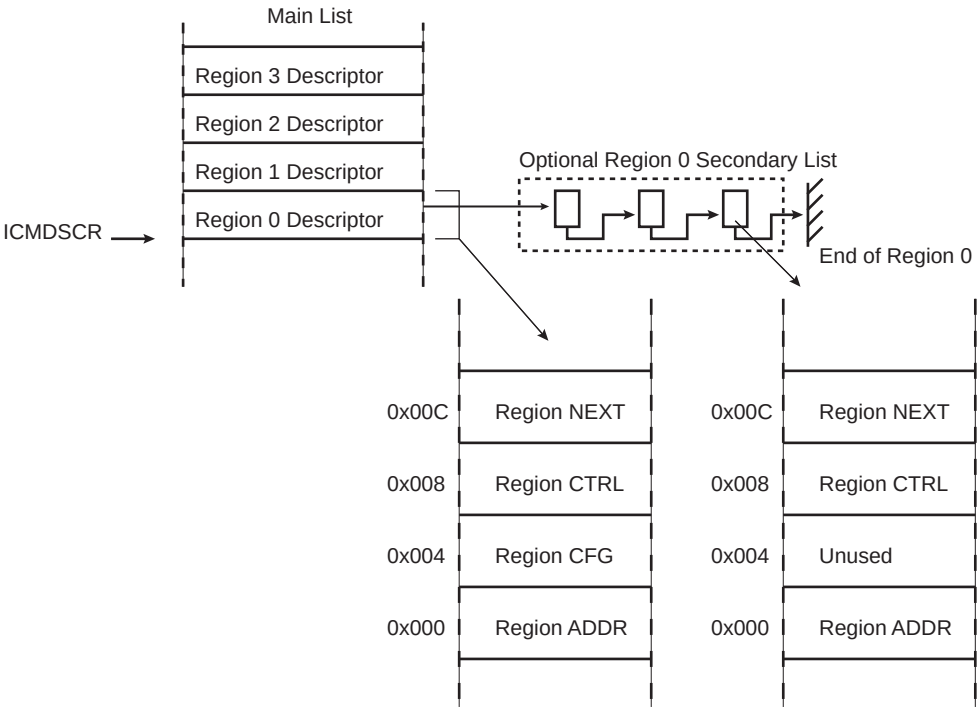
Figure 53-3. ICM Region Descriptor and Hash Areas



Downloaded from Arrow.com.

Each region descriptor supports gathering of data through the use of the Secondary List. Unlike the Main List, the Secondary List cannot modify the configuration attributes of the region. When the end of the Secondary List has been encountered, the ICM returns to the Main List. Memory integrity monitoring can be considered as a background service and the mandatory bandwidth shall be very limited. In order to limit the ICM memory bandwidth, use the BBC field of the ICM_CFG register to control ICM memory load.

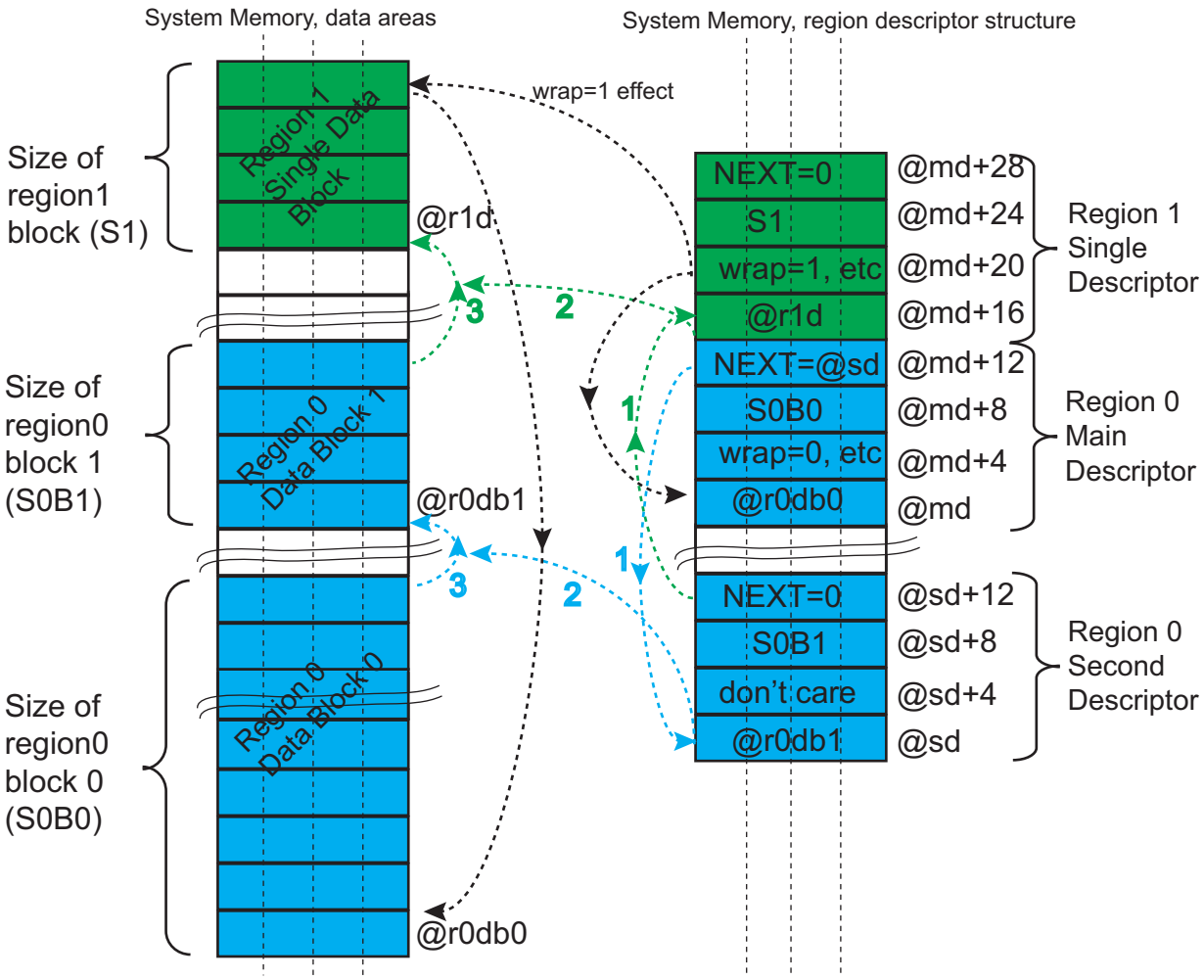
Figure 53-4. Region Descriptor



Downloaded from Arrow.com.

Figure 53-5 shows an example of the mandatory ICM settings to monitor three memory data blocks of the system memory (defined as two regions) with one region being not contiguous (two separate areas) and one contiguous memory area. For each said region, the SHA algorithm may be independently selected (different for each region). The wrap allows continous monitoring.

Figure 53-5. Example: Monitoring of 3 Memory Data Blocks (Defined as 2 Regions)



Downloaded from Arrow.com.

53.5.2 ICM Region Descriptor Structure

The ICM Region Descriptor Area is a contiguous area of system memory that the controller and the processor can access. When the ICM controller is activated, the controller performs a descriptor fetch operation at *(ICM_DSCR) address. If the Main List contains more than one descriptor (i.e., more than one region is to be monitored), the fetch address is *(ICM_DSCR) + (RID<<4) where RID is the region identifier.

Table 53-2. Region Descriptor Structure (Main List)

Offset	Structure Member	Name
ICM_DSCR+0x000+RID*(0x10)	ICM Region Start Address	ICM_RADDR
ICM_DSCR+0x004+RID*(0x10)	ICM Region Configuration	ICM_RCFG
ICM_DSCR+0x008+RID*(0x10)	ICM Region Control	ICM_RCTRL
ICM_DSCR+0x00C+RID*(0x10)	ICM Region Next Address	ICM_RNEXT

Downloaded from Arrow.com.

53.5.2.1 ICM Region Start Address Structure Member

Name: ICM_RADDR

Address: ICM_DSCR+0x000+RID*(0x10)

Access: Read/Write

31	30	29	28	27	26	25	24
RADDR							

23	22	21	20	19	18	17	16
RADDR							

15	14	13	12	11	10	9	8
RADDR							

7	6	5	4	3	2	1	0
RADDR							

- **RADDR: Region Start Address**

This field indicates the first byte address of the region.

Downloaded from Arrow.com.

53.5.2.2 ICM Region Configuration Structure Member

Name: ICM_RCFG

Address: ICM_DSCR+0x004+RID*(0x10)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	ALGO			–	PROCDLY	SUIEN	ECIEN
7	6	5	4	3	2	1	0
WCIEN	BEIEN	DMIEN	RHIEN	–	EOM	WRAP	CDWBN

- **CDWBN: Compare Digest or Write Back Digest**

0: The digest is written to the Hash area.

1: The digest value is compared to the digest stored in the Hash area.

- **WRAP: Wrap Command**

0: The next region descriptor address loaded is the current region identifier descriptor address incremented by 0x10.

1: The next region descriptor address loaded is ICM_DSCR.

- **EOM: End Of Monitoring**

0: The current descriptor does not terminate the monitoring.

1: The current descriptor terminates the Main List. WRAP bit value has no effect.

- **RHIEN: Region Hash Completed Interrupt Disable (Default Enabled)**

0: The ICM_ISR RHC[*i*] flag is set when the field NEXT = 0 in a descriptor of the main or second list.

1: The ICM_ISR RHC[*i*] flag remains cleared even if the setting condition is met.

- **DMIEN: Digest Mismatch Interrupt Disable (Default Enabled)**

0: The ICM_ISR RBE[*i*] flag is set when the hash value just calculated from the processed region differs from expected hash value.

1: The ICM_ISR RBE[*i*] flag remains cleared even if the setting condition is met.

- **BEIEN: Bus Error Interrupt Disable (Default Enabled)**

0: The flag is set when an error is reported on the system bus by the bus MATRIX.

1: The flag remains cleared even if the setting condition is met.

- **WCIEN: Wrap Condition Interrupt Disable (Default Enabled)**

0: The ICM_ISR RWC[*i*] flag is set when the WRAP bit is set in a descriptor of the main list.

1: The ICM_ISR RWC[*i*] flag remains cleared even if the setting condition is met.

Downloaded from Arrow.com.

- **ECIEN: End Bit Condition Interrupt (Default Enabled)**

0: The ICM_ISR REC[*i*] flag is set when the descriptor having the EOM bit set is processed.

1: The ICM_ISR REC[*i*] flag remains cleared even if the setting condition is met.

- **SUIEN: Monitoring Status Updated Condition Interrupt (Default Enabled)**

0: The ICM_ISR RSU[*i*] flag is set when the corresponding descriptor is loaded from memory to ICM.

1: The ICM_ISR RSU[*i*] flag remains cleared even if the setting condition is met.

- **PROCDLY: Processing Delay**

Value	Name	Description
0	SHORTEST	SHA processing runtime is the shortest one
1	LONGEST	SHA processing runtime is the longest one

When SHA1 algorithm is processed, the runtime period is either 85 or 209 clock cycles.

When SHA256 or SHA224 algorithm is processed, the runtime period is either 72 or 194 clock cycles.

- **ALGO: SHA Algorithm**

Value	Name	Description
0	SHA1	SHA1 algorithm processed
1	SHA256	SHA256 algorithm processed
4	SHA224	SHA224 algorithm processed

Values which are not listed in the table must be considered as “reserved”.

Downloaded from Arrow.com.

53.5.2.3 ICM Region Control Structure Member

Name: ICM_RCTRL

Address: ICM_DSCR+0x008+RID*(0x10)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
TRSIZE							
7	6	5	4	3	2	1	0
TRSIZE							

- **TRSIZE: Transfer Size for the Current Chunk of Data**

ICM performs a transfer of (TRSIZE + 1) blocks of 512 bits.

Downloaded from Arrow.com.

53.5.2.4 ICM Region Next Address Structure Member

Name: ICM_RNEXT

Address: ICM_DSCR+0x00C+RID*(0x10)

Access: Read/Write

31	30	29	28	27	26	25	24
NEXT							

23	22	21	20	19	18	17	16
NEXT							

15	14	13	12	11	10	9	8
NEXT							

7	6	5	4	3	2	1	0
NEXT						–	–

- **NEXT: Region Transfer Descriptor Next Address**

When configured to 0, this field indicates that the current descriptor is the last descriptor of the Secondary List, otherwise it points at a new descriptor of the Secondary List.

Downloaded from Arrow.com.

53.5.3 ICM Hash Area

The ICM Hash Area is a contiguous area of system memory that the controller and the processor can access. The physical location is configured in the ICM hash area start address register. This address is a multiple of 128 bytes. If the CDWBN bit of the context register is cleared (i.e., Write Back activated), the ICM controller performs a digest write operation at the following starting location: *(ICM_HASH) + (RID<<5), where RID is the current region context identifier. If the CDWBN bit of the context register is set (i.e., Digest Comparison activated), the ICM controller performs a digest read operation at the same address.

53.5.3.1 Message Digest Example

Considering the following **512 bits message** (example given in FIPS 180-2):

“61626380000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000018”

The message is written to memory in a Little Endian (LE) system architecture.

Table 53-3. 512 bits Message Memory Mapping

	Address Offset / Byte Lane			
Memory Address	**0x3 / 31:24**	**0x2 / 23:16**	**0x1 / 15:8**	**0x0 / 7:0**
0x000	80	63	62	61
0x004–0x038	00	00	00	00
0x03C	18	00	00	00

The digest is stored at the memory location pointed at by the ICM_HASH pointer with a Region Offset.

Table 53-4. LE Resulting SHA-160 Message Digest Memory Mapping

	Address Offset / Byte Lane			
Memory Address	**0x3 / 31:24**	**0x2 / 23:16**	**0x1 / 15:8**	**0x0 / 7:0**
0x000	36	3e	99	a9
0x004	6a	81	06	47
0x008	71	25	3e	ba
0x00C	6c	c2	50	78
0x010	9d	d8	d0	9c

Table 53-5. Resulting SHA-224 Message Digest Memory Mapping

	Address Offset / Byte Lane			
Memory Address	**0x3 / 31:24**	**0x2 / 23:16**	**0x1 / 15:8**	**0x0 / 7:0**
0x000	22	7d	09	23
0x004	22	d8	05	34
0x008	77	a4	42	86
0x00C	b3	55	a2	bd
0x010	e4	bc	ad	2a
0x014	f7	b3	a0	bd
0x018	a7	9d	6c	e3

Downloaded from Arrow.com.

Table 53-6. Resulting SHA-256 Message Digest Memory Mapping

Memory Address	Address Offset / Byte Lane			
	0x3 / 31:24	0x2 / 23:16	0x1 / 15:8	0x0 / 7:0
0x000	bf	16	78	ba
0x004	ea	cf	01	8f
0x008	de	40	41	41
0x00C	23	22	ae	5d
0x010	a3	61	03	b0
0x014	9c	7a	17	96
0x018	61	ff	10	b4
0x01C	ad	15	00	f2

Considering the following **1024 bits message** (example given in FIPS 180-2):

“61626380000000000

The bits RHIEN or ECIEN must be written to 1 in the region descriptor structure member ICM_RCTRL. The flag RHC[*i*], *i* being the region index, is set (if RHIEN is set) when the hash result is available at address defined in ICM_HASH. The flag REC[*i*], *i* being the region index, is set (if ECIEN is set) when the hash result is available at the address defined in ICM_HASH.

An interrupt is generated if the bit RHC[*i*] is written to 1 in the ICM_IER (if RHC[*i*] is set in ICM_RCTRL of region *i*) or if the bit REC[*i*] is written to 1 in the ICM_IER (if REC[*i*] is set in ICM_RCTRL of region *i*).

53.5.4.2 Processing Period

The SHA engine processing period can be configured.

The short processing period allows to allocate bandwidth to the SHA module whereas the long processing period allocates more bandwidth on the system bus to other applications.

In SHA mode, the shortest processing period is 85 clock cycles + 2 clock cycles for start command synchronization. The longest period is 209 clock cycles + 2 clock cycles.

In SHA256 and SHA224 modes, the shortest processing period is 72 clock cycles + 2 clock cycles for start command synchronization. The longest period is 194 clock cycles + 2 clock cycles.

53.5.5 ICM Automatic Monitoring Mode

The ASCD bit of the ICM_CFG register is used to activate the ICM Automatic Mode. When ICM_CFG.ASCD is set, the ICM performs the following actions:

- The ICM controller passes through the Main List once with CDWBN bit in the context register at 0 (i.e., Write Back activated) and EOM bit in context register at 0.
- When WRAP = 1 in ICM_RCFG, the ICM controller enters active monitoring with CDWBN bit in context register now set and EOM bit in context register cleared. Bits CDWBN and EOM in ICM_RCFG have no effect.

53.5.6 Programming the ICM for Multiple Regions

Table 53-8. Region Attributes

Transfer Type		Main List	ICM_RCFG			ICM_RNEXT	Comments
			CDWBN	WRAP	EOM	NEXT	
Single Region	Contiguous list of blocks Digest written to memory Monitoring disabled	1 item	0	0	1	0	The Main List contains only one descriptor. The Secondary List is empty for that descriptor. The digest is computed and saved to memory.
	Non-contiguous list of blocks Digest written to memory Monitoring disabled	1 item	0	0	1	Secondary List address of the current region identifier	The Main List contains only one descriptor. The Secondary List describes the layout of the non-contiguous region.
	Contiguous list of blocks Digest comparison enabled Monitoring enabled	1 item	1	1	0	0	When the hash computation is terminated, the digest is compared with the one saved in memory.

Downloaded from Arrow.com.

Table 53-8. Region Attributes

Transfer Type		Main List	ICM_RCFG			ICM_RNEXT	Comments
			CDWBN	WRAP	EOM	NEXT	
Multiple Regions	Contiguous list of blocks Digest written to memory Monitoring disabled	More than one item	0	0	1 for the last, 0 otherwise	0	ICM passes through the list once.
	Contiguous list of blocks Digest comparison is enabled Monitoring is enabled	More than one item	1	1 for the last, 0 otherwise	0	0	ICM performs active monitoring of the regions. If a mismatch occurs, an interrupt is raised.
	Non-contiguous list of blocks Digest is written to memory Monitoring is disabled	More than one item	0	0	1	Secondary List address	ICM performs hashing and saves digests to the Hash area.
	Non-contiguous list of blocks Digest comparison is enabled Monitoring is enabled	More than one item	1	1	0	Secondary List address	ICM performs data gathering on a per region basis.

53.5.7 Security Features

When an undefined register access occurs, the URAD bit in the Interrupt Status Register (ICM_ISR) is set if unmasked. Its source is then reported in the Undefined Access Status Register (ICM_UASR). Only the first undefined register access is available through the ICM_UASR.URAT field.

Several kinds of unspecified register accesses can occur:

- Unspecified structure member set to one detected when the descriptor is loaded
- Configuration register (ICM_CFG) modified during active monitoring
- Descriptor register (ICM_DSCR) modified during active monitoring
- Hash register (ICM_HASH) modified during active monitoring
- Write-only register read access

The URAD bit and the URAT field can only be reset by writing a 1 to the ICM_CTRL.SWRST bit.

Downloaded from Arrow.com.

53.6 Integrity Check Monitor (ICM) User Interface

Table 53-9. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Configuration Register	ICM_CFG	Read/Write	0x0
0x04	Control Register	ICM_CTRL	Write-only	–
0x08	Status Register	ICM_SR	Write-only	–
0x0C	Reserved	–	–	–
0x10	Interrupt Enable Register	ICM_IER	Write-only	–
0x14	Interrupt Disable Register	ICM_IDR	Write-only	–
0x18	Interrupt Mask Register	ICM_IMR	Read-only	0x0
0x1C	Interrupt Status Register	ICM_ISR	Read-only	0x0
0x20	Undefined Access Status Register	ICM_UASR	Read-only	0x0
0x24–0x2C	Reserved	–	–	–
0x30	Region Descriptor Area Start Address Register	ICM_DSCR	Read/Write	0x0
0x34	Region Hash Area Start Address Register	ICM_HASH	Read/Write	0x0
0x38	User Initial Hash Value 0 Register	ICM_UIHVAL0	Write-only	–
...	...	...	...	...
0x54	User Initial Hash Value 7	ICM_UIHVAL7	Write-only	–
0x58–0xFC	Reserved	–	–	–

Downloaded from Arrow.com.

53.6.1 ICM Configuration Register

Name: ICM_CFG

Address: 0x40048000

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
UALGO			UIHASH	–	–	DUALBUFF	ASCD
7	6	5	4	3	2	1	0
BBC				–	SLBDIS	EOMDIS	WBDIS

- **WBDIS: Write Back Disable**

0: Write Back Operations are permitted.

1: Write Back Operations are forbidden. Context register CDWBN bit is internally set to one and cannot be modified by a linked list element. The CDWBN bit of the ICM_RCFG structure member has no effect.

When ASCD bit of the ICM_CFG register is set, WBDIS bit value has no effect.

- **EOMDIS: End of Monitoring Disable**

0: End of Monitoring is permitted

1: End of Monitoring is forbidden. The EOM bit of the ICM_RCFG structure member has no effect.

- **SLBDIS: Secondary List Branching Disable**

0: Branching to the Secondary List is permitted.

1: Branching to the Secondary List is forbidden. The NEXT field of the ICM_RNEXT structure member has no effect and is always considered as zero.

- **BBC: Bus Burden Control**

This field is used to control the burden of the ICM system bus. The number of system clock cycles between the end of the current processing and the next block transfer is set to 2^{BBC}. Up to 32,768 cycles can be inserted.

- **ASCD: Automatic Switch To Compare Digest**

0: Automatic mode is disabled.

1: When this mode is enabled, the ICM controller automatically switches to active monitoring after the first Main List pass. Both CDWBN and WBDIS bits have no effect. A one must be written to the EOM bit in ICM_RCFG to terminate the monitoring.

- **DUALBUFF: Dual Input Buffer**

0: Dual Input buffer mode is disabled.

1: Dual Input buffer mode is enabled (Better performances, higher bandwidth required on system bus).

Downloaded from Arrow.com.

- **UIHASH: User Initial Hash Value**

0: The secure hash standard provides the initial hash value.

1: The initial hash value is programmable. Field UALGO provides the SHA algorithm. The ALGO field of the ICM_RCFG structure member has no effect.

- **UALGO: User SHA Algorithm**

Value	Name	Description
0	SHA1	SHA1 algorithm processed
1	SHA256	SHA256 algorithm processed
4	SHA224	SHA224 algorithm processed

Downloaded from Arrow.com.

53.6.2 ICM Control Register

Name: ICM_CTRL

Address: 0x40048004

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
RMEN				RMDIS			
7	6	5	4	3	2	1	0
REHASH				–	SWRST	DISABLE	ENABLE

- **ENABLE: ICM Enable**

0: No effect

1: When set to one, the ICM controller is activated.

- **DISABLE: ICM Disable Register**

0: No effect

1: The ICM controller is disabled. If a region is active, this region is terminated.

- **SWRST: Software Reset**

0: No effect

1: Resets the ICM controller.

- **REHASH: Recompute Internal Hash**

0: No effect

1: When REHASH[*i*] is set to one, Region *i* digest is re-computed. This bit is only available when region monitoring is disabled.

- **RMDIS: Region Monitoring Disable**

0: No effect

1: When bit RMDIS[*i*] is set to one, the monitoring of region with identifier *i* is disabled.

- **RMEN: Region Monitoring Enable**

0: No effect

1: When bit RMEN[*i*] is set to one, the monitoring of region with identifier *i* is activated.

Monitoring is activated by default.

Downloaded from Arrow.com.

53.6.3 ICM Status Register

Name: ICM_SR

Address: 0x40048008

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
RMDIS				RAWRMDIS			
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	ENABLE

- **ENABLE: ICM Controller Enable Register**

0: ICM controller is disabled

1: ICM controller is activated

- **RAWRMDIS: Region Monitoring Disabled Raw Status**

0: Region *i* monitoring has been activated by writing a 1 in RMEN[*i*] of ICM_CTRL

1: Region *i* monitoring has been deactivated by writing a 1 in RMDIS[*i*] of ICM_CTRL

- **RMDIS: Region Monitoring Disabled Status**

0: Region *i* is being monitored (occurs after integrity check value has been calculated and written to Hash area)

1: Region *i* monitoring is not being monitored

Downloaded from Arrow.com.

53.6.4 ICM Interrupt Enable Register

Name: ICM_IER

Address: 0x40048010

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	URAD

23	22	21	20	19	18	17	16
RSU				REC			

15	14	13	12	11	10	9	8
RWC				RBE			

7	6	5	4	3	2	1	0
RDM				RHC			

- **RHC: Region Hash Completed Interrupt Enable**

0: No effect

1: When RHC[*i*] is set to one, the Region *i* Hash Completed interrupt is enabled.

- **RDM: Region Digest Mismatch Interrupt Enable**

0: No effect

1: When RDM[*i*] is set to one, the Region *i* Digest Mismatch interrupt is enabled.

- **RBE: Region Bus Error Interrupt Enable**

0: No effect

1: When RBE[*i*] is set to one, the Region *i* Bus Error interrupt is enabled.

- **RWC: Region Wrap Condition detected Interrupt Enable**

0: No effect

1: When RWC[*i*] is set to one, the Region *i* Wrap Condition interrupt is enabled.

- **REC: Region End bit Condition Detected Interrupt Enable**

0: No effect

1: When REC[*i*] is set to one, the region *i* End bit Condition interrupt is enabled.

- **RSU: Region Status Updated Interrupt Disable**

0: No effect

1: When RSU[*i*] is set to one, the region *i* Status Updated interrupt is enabled.

- **URAD: Undefined Register Access Detection Interrupt Enable**

0: No effect

1: The Undefined Register Access interrupt is enabled.

Downloaded from Arrow.com.

53.6.5 ICM Interrupt Disable Register

Name: ICM_IDR

Address: 0x40048014

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	URAD

23	22	21	20	19	18	17	16
RSU				REC			

15	14	13	12	11	10	9	8
RWC				RBE			

7	6	5	4	3	2	1	0
RDM				RHC			

- **RHC: Region Hash Completed Interrupt Disable**

0: No effect

1: When RHC[*i*] is set to one, the Region *i* Hash Completed interrupt is disabled.

- **RDM: Region Digest Mismatch Interrupt Disable**

0: No effect

1: When RDM[*i*] is set to one, the Region *i* Digest Mismatch interrupt is disabled.

- **RBE: Region Bus Error Interrupt Disable**

0: No effect

1: When RBE[*i*] is set to one, the Region *i* Bus Error interrupt is disabled.

- **RWC: Region Wrap Condition Detected Interrupt Disable**

0: No effect

1: When RWC[*i*] is set to one, the Region *i* Wrap Condition interrupt is disabled.

- **REC: Region End bit Condition detected Interrupt Disable**

0: No effect

1: When REC[*i*] is set to one, the region *i* End bit Condition interrupt is disabled.

- **RSU: Region Status Updated Interrupt Disable**

0: No effect

1: When RSU[*i*] is set to one, the region *i* Status Updated interrupt is disabled.

- **URAD: Undefined Register Access Detection Interrupt Disable**

0: No effect

1: Undefined Register Access Detection interrupt is disabled.

Downloaded from Arrow.com.

53.6.6 ICM Interrupt Mask Register

Name: ICM_IMR

Address: 0x40048018

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	URAD
23	22	21	20	19	18	17	16
RSU				REC			
15	14	13	12	11	10	9	8
RWC				RBE			
7	6	5	4	3	2	1	0
RDM				RHC			

- **RHC: Region Hash Completed Interrupt Mask**

0: When RHC[*i*] is set to zero, the interrupt is disabled for region i.

1: When RHC[*i*] is set to one, the interrupt is enabled for region i.

- **RDM: Region Digest Mismatch Interrupt Mask**

0: When RDM[*i*] is set to zero, the interrupt is disabled for region i.

1: When RDM[*i*] is set to one, the interrupt is enabled for region i.

- **RBE: Region Bus Error Interrupt Mask**

0: When RBE[*i*] is set to zero, the interrupt is disabled for region i.

1: When RBE[*i*] is set to one, the interrupt is enabled for region i.

- **RWC: Region Wrap Condition Detected Interrupt Mask**

0: When RWC[*i*] is set to zero, the interrupt is disabled for region i.

1: When RWC[*i*] is set to one, the interrupt is enabled for region i.

- **REC: Region End bit Condition Detected Interrupt Mask**

0: When REC[*i*] is set to zero, the interrupt is disabled for region i.

1: When REC[*i*] is set to one, the interrupt is enabled for region i.

- **RSU: Region Status Updated Interrupt Mask**

0: When RSU[*i*] is set to zero, the interrupt is disabled for region i.

1: When RSU[*i*] is set to one, the interrupt is enabled for region i.

- **URAD: Undefined Register Access Detection Interrupt Mask**

0: Interrupt is disabled

1: Interrupt is enabled.

Downloaded from Arrow.com.

53.6.7 ICM Interrupt Status Register

Name: ICM_ISR

Address: 0x4004801C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	URAD

23	22	21	20	19	18	17	16
RSU				REC			

15	14	13	12	11	10	9	8
RWC				RBE			

7	6	5	4	3	2	1	0
RDM				RHC			

- **RHC: Region Hash Completed**

When RHC[*i*] is set, it indicates that the ICM has completed the region with identifier *i*.

- **RDM: Region Digest Mismatch**

When RDM[*i*] is set, it indicates that there is a digest comparison mismatch between the hash value of the region with identifier *i* and the reference value located in the Hash Area.

- **RBE: Region Bus Error**

When RBE[*i*] is set, it indicates that a bus error has been detected while hashing memory region *i*.

- **RWC: Region Wrap Condition Detected**

When RWC[*i*] is set, it indicates that a wrap condition has been detected.

- **REC: Region End bit Condition Detected**

When REC[*i*] is set, it indicates that an end bit condition has been detected.

- **RSU: Region Status Updated Detected**

When RSU[*i*] is set, it indicates that a region status updated condition has been detected.

- **URAD: Undefined Register Access Detection Status**

0: No undefined register access has been detected since the last SWRST.

1: At least one undefined register access has been detected since the last SWRST.

The URAD bit is only reset by the SWRST bit in the ICM_CTRL register.

The URAT field in the ICM_UASR indicates the unspecified access type.

Downloaded from Arrow.com.

53.6.8 ICM Undefined Access Status Register

Name: ICM_UASR

Address: 0x40048020

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	URAT		

- **URAT: Undefined Register Access Trace**

Value	Name	Description
0	UNSPEC_STRUCT_MEMBER	Unspecified structure member set to one detected when the descriptor is loaded.
1	ICM_CFG_MODIFIED	ICM_CFG modified during active monitoring.
2	ICM_DSCR_MODIFIED	ICM_DSCR modified during active monitoring.
3	ICM_HASH_MODIFIED	ICM_HASH modified during active monitoring
4	READ_ACCESS	Write-only register read access

Only the first Undefined Register Access Trace is available through the URAT field.

The URAT field is only reset by the SWRST bit in the ICM_CTRL register.

Downloaded from Arrow.com.

53.6.9 ICM Descriptor Area Start Address Register

Name: ICM_DSCR

Address: 0x40048030

Access: Read/Write

31	30	29	28	27	26	25	24
DASA							

23	22	21	20	19	18	17	16
DASA							

15	14	13	12	11	10	9	8
DASA							

7	6	5	4	3	2	1	0
DASA		–	–	–	–	–	–

- **DASA: Descriptor Area Start Address**

The start address is a multiple of the total size of the data structure (64 bytes).

Downloaded from Arrow.com.

53.6.10 ICM Hash Area Start Address Register

Name: ICM_HASH

Address: 0x40048034

Access: Read/Write

31	30	29	28	27	26	25	24
HASA							
23	22	21	20	19	18	17	16
HASA							
15	14	13	12	11	10	9	8
HASA							
7	6	5	4	3	2	1	0
HASA	–	–	–	–	–	–	–

- **HASA: Hash Area Start Address**

This field points at the Hash memory location. The address must be a multiple of 128 bytes.

Downloaded from Arrow.com.

53.6.11 ICM User Initial Hash Value Register

Name: ICM_UIHVALx [x=0..7]

Address: 0x40048038

Access: Write-only

31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

- **VAL: Initial Hash Value**

When UIHASH bit of IMC_CFG register is set, the Initial Hash Value is user-programmable.

To meet the desired standard, use the following example values.

For ICM_UIHVAL0 field:

Example	Comment
0x67452301	SHA1 algorithm
0xC1059ED8	SHA224 algorithm
0x6A09E667	SHA256 algorithm

For ICM_UIHVAL1 field:

Example	Comment
0xEFCDAB89	SHA1 algorithm
0x367CD507	SHA224 algorithm
0xBB67AE85	SHA256 algorithm

For ICM_UIHVAL2 field:

Example	Comment
0x98BADCFE	SHA1 algorithm
0x3070DD17	SHA224 algorithm
0x3C6EF372	SHA256 algorithm

For ICM_UIHVAL3 field:

Example	Comment
0x10325476	SHA1 algorithm
0xF70E5939	SHA224 algorithm
0xA54FF53A	SHA256 algorithm

Downloaded from Arrow.com.

For ICM_UIHVAL4 field:

Example	Comment
0xC3D2E1F0	SHA1 algorithm
0xFFC00B31	SHA224 algorithm
0x510E527F	SHA256 algorithm

For ICM_UIHVAL5 field:

Example	Comment
0x68581511	SHA224 algorithm
0x9B05688C	SHA256 algorithm

For ICM_UIHVAL6 field:

Example	Comment
0x64F98FA7	SHA224 algorithm
0x1F83D9AB	SHA256 algorithm

For ICM_UIHVAL7 field:

Example	Comment
0xBEFA4FA4	SHA224 algorithm
0x5BE0CD19	SHA256 algorithm

Example of Initial Value for SHA-1 Algorithm

Register Address	Address Offset / Byte Lane			
	0x3 / 31:24	0x2 / 23:16	0x1 / 15:8	0x0 / 7:0
0x000 ICM_UIHVAL0	01	23	45	67
0x004 ICM_UIHVAL1	89	ab	cd	ef
0x008 ICM_UIHVAL2	fe	dc	ba	98
0x00C ICM_UIHVAL3	76	54	32	10
0x010 ICM_UIHVAL4	f0	e1	d2	c3

Downloaded from Arrow.com.

54. True Random Number Generator (TRNG)

54.1 Description

The True Random Number Generator (TRNG) passes the American *NIST Special Publication 800-22* (*A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*) and the *Diehard Suite of Tests*.

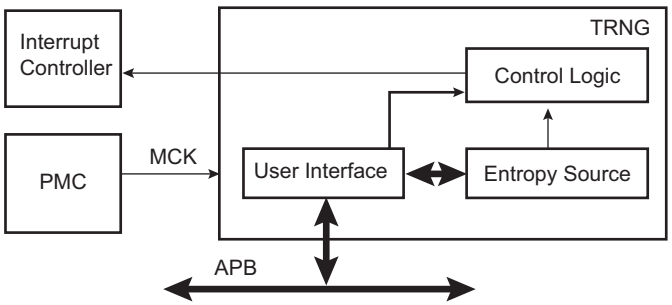
The TRNG may be used as an entropy source for seeding an NIST approved DRNG (Deterministic RNG) as required by FIPS PUB 140-2 and 140-3.

54.2 Embedded Characteristics

- Passes *NIST Special Publication 800-22* Test Suite
- Passes *Diehard Suite of Tests*
- May be used as Entropy Source for seeding an NIST approved DRNG (Deterministic RNG) as required by FIPS PUB 140-2 and 140-3
- Provides a 32-bit Random Number Every 84 Clock Cycles

54.3 Block Diagram

Figure 54-1. TRNG Block Diagram



54.4 Product Dependencies

54.4.1 Power Management

The TRNG interface may be clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the TRNG user interface clock. The user interface clock is independent from any clock that may be used in the entropy source logic circuitry. The source of entropy can be enabled before enabling the user interface clock.

54.4.2 Interrupt Sources

The TRNG interface has an interrupt line connected to the Interrupt Controller. In order to handle interrupts, the Interrupt Controller must be programmed before configuring the TRNG.

Table 54-1. Peripheral IDs

Instance	ID
TRNG	57

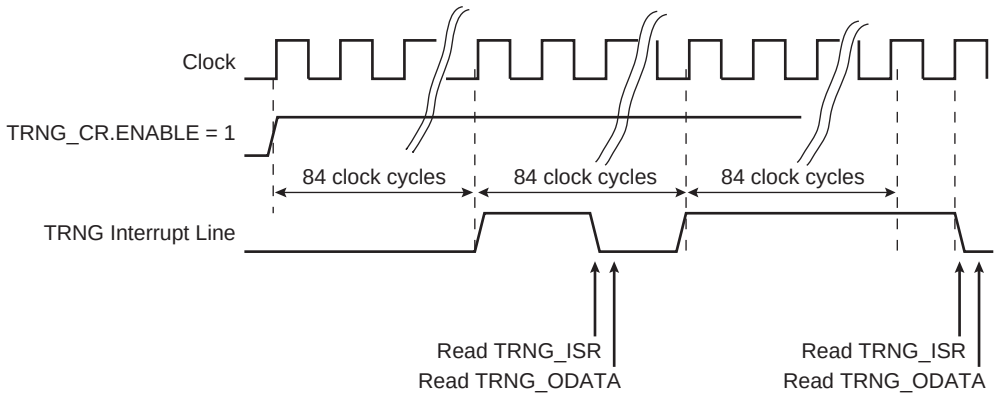
Downloaded from Arrow.com.

54.5 Functional Description

As soon as the TRNG is enabled in the control register (TRNG_CR), the generator provides one 32-bit value every 84 clock cycles. The TRNG interrupt line can be enabled in the TRNG_IER (respectively disabled in the TRNG_IDR). This interrupt is set when a new random value is available and is cleared when the status register (TRNG_ISR) is read. The flag DATRDY of the (TRNG_ISR) is set when the random data is ready to be read out on the 32-bit output data register (TRNG_ODATA).

The normal mode of operation checks that the status register flag equals 1 before reading the output data register when a 32-bit random value is required by the software application.

Figure 54-2. TRNG Data Generation Sequence



Downloaded from Arrow.com.

54.6 True Random Number Generator (TRNG) User Interface

Table 54-2. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	TRNG_CR	Write-only	_
0x04–0x0C	Reserved	–	_	–
0x10	Interrupt Enable Register	TRNG_IER	Write-only	–
0x14	Interrupt Disable Register	TRNG_IDR	Write-only	–
0x18	Interrupt Mask Register	TRNG_IMR	Read-only	0x0000_0000
0x1C	Interrupt Status Register	TRNG_ISR	Read-only	0x0000_0000
0x20–0x4C	Reserved	–	_	–
0x50	Output Data Register	TRNG_ODATA	Read-only	0x0000_0000
0x54–0xFC	Reserved	–	_	–

Downloaded from Arrow.com.

54.6.1 TRNG Control Register

Name: TRNG_CR

Address: 0x40070000

Access: Write-only

31	30	29	28	27	26	25	24
KEY							
23	22	21	20	19	18	17	16
KEY							
15	14	13	12	11	10	9	8
KEY							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	ENABLE

- **ENABLE: Enables the TRNG to Provide Random Values**

0: Disables the TRNG.

1: Enables the TRNG if 0x524E47 (“RNG” in ASCII) is written in KEY field at the same time.

- **KEY: Security Key**

Value	Name	Description
0x524E47	PASSWD	Writing any other value in this field aborts the write operation.

Downloaded from Arrow.com.

54.6.2 TRNG Interrupt Enable Register

Name: TRNG_IER

Address: 0x40070010

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

- **DATRDY: Data Ready Interrupt Enable**

0: No effect.

1: Enables the corresponding interrupt.

Downloaded from Arrow.com.

54.6.3 TRNG Interrupt Disable Register

Name: TRNG_IDR

Address: 0x40070014

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

- **DATRDY: Data Ready Interrupt Disable**

0: No effect.

1: Disables the corresponding interrupt.

Downloaded from Arrow.com.

54.6.4 TRNG Interrupt Mask Register

Name: TRNG_IMR

Address: 0x40070018

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

- **DATRDY: Data Ready Interrupt Mask**

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

Downloaded from Arrow.com.

54.6.5 TRNG Interrupt Status Register

Name: TRNG_ISR

Address: 0x4007001C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
		–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

- **DATRDY: Data Ready**

0: Output data is not valid or TRNG is disabled.

1: New random value is completed.

DATRDY is cleared when this register is read.

Downloaded from Arrow.com.

54.6.6 TRNG Output Data Register

Name: TRNG_ODATA

Address: 0x40070050

Access: Read-only

31	30	29	28	27	26	25	24
ODATA							

23	22	21	20	19	18	17	16
ODATA							

15	14	13	12	11	10	9	8
ODATA							

7	6	5	4	3	2	1	0
ODATA							

- **ODATA: Output Data**

The 32-bit Output Data register contains the 32-bit random data.

Downloaded from Arrow.com.

55. Advanced Encryption Standard (AES)

55.1 Description

The Advanced Encryption Standard (AES) is compliant with the American *FIPS (Federal Information Processing Standard) Publication 197* specification.

The AES supports all five confidentiality modes of operation for symmetrical key block cipher algorithms (ECB, CBC, OFB, CFB and CTR), as specified in the *NIST Special Publication 800-38A Recommendation,* as well as Galois/Counter Mode (GCM) as specified in the *NIST Special Publication 800-38D Recommendation*. It is compatible with all these modes via DMA Controller channels, minimizing processor intervention for large buffer transfers.

The 128-bit/192-bit/256-bit key is stored in four/six/eight 32-bit write-only AES Key Word Registers (AES_KEYWR0–3).

The 128-bit input data and initialization vector (for some modes) are each stored in four 32-bit write-only AES Input Data Registers (AES_IDATAR0–3) and AES Initialization Vector Registers (AES_IVR0–3).

As soon as the initialization vector, the input data and the key are configured, the encryption/decryption process may be started. Then the encrypted/decrypted data are ready to be read out on the four 32-bit AES Output Data Registers (AES_ODATAR0–3) or through the DMA channels.

55.2 Embedded Characteristics

- Compliant with *FIPS Publication 197, Advanced Encryption Standard (AES)*
- 128-bit/192-bit/256-bit Cryptographic Key
- 12/14/16 Clock Cycles Encryption/Decryption Processing Time with a 128-bit/192-bit/256-bit Cryptographic Key
- Double Input Buffer Optimizes Runtime
- Support of the Modes of Operation Specified in the *NIST Special Publication 800-38A* and *NIST Special Publication 800-38D*:
  - Electronic Code Book (ECB)
  - Cipher Block Chaining (CBC) including CBC-MAC
  - Cipher Feedback (CFB)
  - Output Feedback (OFB)
  - Counter (CTR)
  - Galois/Counter Mode (GCM)
- 8, 16, 32, 64 and 128-bit Data Sizes Possible in CFB Mode
- Last Output Data Mode Allows Optimized Message Authentication Code (MAC) Generation
- Connection to DMA Optimizes Data Transfers for all Operating Modes

Downloaded from Arrow.com.

55.3 Product Dependencies

55.3.1 Power Management

The AES may be clocked through the Power Management Controller (PMC), so the programmer must first to configure the PMC to enable the AES clock.

55.3.2 Interrupt Sources

The AES interface has an interrupt line connected to the Interrupt Controller.

Handling the AES interrupt requires programming the Interrupt Controller before configuring the AES.

Table 55-1. Peripheral IDs

Instance	ID
AES	56

Downloaded from Arrow.com.

55.4 Functional Description

The Advanced Encryption Standard (AES) specifies a FIPS-approved cryptographic algorithm that can be used to protect electronic data. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information.

Encryption converts data to an unintelligible form called ciphertext. Decrypting the ciphertext converts the data back into its original form, called plaintext. The CIPHER bit in the AES Mode Register (AES_MR) allows selection between the encryption and the decryption processes.

The AES is capable of using cryptographic keys of 128/192/256 bits to encrypt and decrypt data in blocks of 128 bits. This 128-bit/192-bit/256-bit key is defined in the AES_KEYWRx.

The input to the encryption processes of the CBC, CFB, and OFB modes includes, in addition to the plaintext, a 128-bit data block called the initialization vector (IV), which must be set in the AES_IVRx. The initialization vector is used in an initial step in the encryption of a message and in the corresponding decryption of the message. The AES_IVRx are also used by the CTR mode to set the counter value.

55.4.1 AES Register Endianness

In ARM processor-based products, the system bus and processors manipulate data in little-endian form. The AES interface requires little-endian format words. However, in accordance with the protocol of the FIPS 197 specification, data is collected, processed and stored by the AES algorithm in big-endian form.

The following example illustrates how to configure the AES:

If the first 64 bits of a message (according to FIPS 197, i.e., big-endian format) to be processed is 0xcafedeca_01234567, then the AES_IDATAR0 and AES_IDATAR1 registers must be written with the following pattern:

- AES_IDATAR0 = 0xcadefeca
- AES_IDATAR1 = 0x67452301

55.4.2 Operation Modes

The AES supports the following modes of operation:

- ECB: Electronic Code Book
- CBC: Cipher Block Chaining
- OFB: Output Feedback
- CFB: Cipher Feedback
  - CFB8 (CFB where the length of the data segment is 8 bits)
  - CFB16 (CFB where the length of the data segment is 16 bits)
  - CFB32 (CFB where the length of the data segment is 32 bits)
  - CFB64 (CFB where the length of the data segment is 64 bits)
  - CFB128 (CFB where the length of the data segment is 128 bits)
- CTR: Counter
- GCM: Galois/Counter Mode

The data pre-processing, post-processing and data chaining for the concerned modes are automatically performed. Refer to the *NIST Special Publication 800-38A* and *NIST Special Publication 800-38D* for more complete information.

These modes are selected by setting the OPMOD field in the AES_MR.

In CFB mode, five data sizes are possible (8, 16, 32, 64 or 128 bits), configurable by means of the CFBS field in the AES_MR (Section 55.5.2 “AES Mode Register”).

Downloaded from Arrow.com.

In CTR mode, the size of the block counter embedded in the module is 16 bits. Therefore, there is a rollover after processing 1 megabyte of data. If the file to be processed is greater than 1 megabyte, this file must be split into fragments of 1 megabyte or less for the first fragment if the initial value of the counter is greater than 0. Prior to loading the first fragment into AES_IDATARx, AES_IVRx must be fully programmed with the initial counter value. For any fragment, after the transfer is completed and prior to transferring the next fragment, AES_IVRx must be programmed with the appropriate counter value.

If the initial value of the counter is greater than 0 and the data buffer size to be processed is greater than 1 megabyte, the size of the first fragment to be processed must be 1 megabyte minus 16x(initial value) to prevent a rollover of the internal 1-bit counter.

To have a sequential increment, the counter value must be programmed with the value programmed for the previous fragment + 2^{16} (or less for the first fragment).

All AES_IVRx fields must be programmed to take into account the possible carry propagation.

55.4.3 Double Input Buffer

The AES_IDATARx can be double-buffered to reduce the runtime of large files.

This mode allows writing a new message block when the previous message block is being processed. This is only possible when DMA accesses are performed (SMOD = 0x2).

The DUALBUFF bit in the AES_MR must be set to ‘1’ to access the double buffer.

55.4.4 Start Modes

The SMOD field in the AES_MR allows selection of the encryption (or decryption) Start mode.

55.4.4.1 Manual Mode

The sequence order is as follows:

1. Write the AES_MR with all required fields, including but not limited to SMOD and OPMOD.
2. Write the 128-bit/192-bit/256-bit key in the AES_KEYWRx.
3. Write the initialization vector (or counter) in the AES_IVRx.

Note: The AES_IVRx concern all modes except ECB.

4. Set the bit DATRDY (Data Ready) in the AES Interrupt Enable Register (AES_IER), depending on whether an interrupt is required or not at the end of processing.
5. Write the data to be encrypted/decrypted in the authorized AES_IDATARx (see Table 55-2).
6. Set the START bit in the AES Control Register (AES_CR) to begin the encryption or the decryption process.
7. When processing completes, the DATRDY flag in the AES Interrupt Status Register (AES_ISR) is raised. If an interrupt has been enabled by setting the DATRDY bit in the AES_IER, the interrupt line of the AES is activated.
8. When software reads one of the AES_ODATARx, the DATRDY bit is automatically cleared.

Table 55-2. Authorized Input Data Registers

Operation Mode	Input Data Registers to Write
ECB	All
CBC	All
OFB	All
128-bit CFB	All
64-bit CFB	AES_IDATAR0 and AES_IDATAR1
32-bit CFB	AES_IDATAR0

Downloaded from Arrow.com.

Table 55-2. Authorized Input Data Registers

Operation Mode	Input Data Registers to Write
16-bit CFB	AES_IDATAR0
8-bit CFB	AES_IDATAR0
CTR	All
GCM	All

Note: In 64-bit CFB mode, writing to AES_IDATAR2 and AES_IDATAR3 is not allowed and may lead to errors in processing.

Note: In 32, 16, and 8-bit CFB modes, writing to AES_IDATAR1, AES_IDATAR2 and AES_IDATAR3 is not allowed and may lead to errors in processing.

55.4.4.2 Auto Mode

The Auto Mode is similar to the manual one, except that in this mode, as soon as the correct number of AES_IDATARx is written, processing is automatically started without any action in the AES_CR.

55.4.4.3 DMA Mode

The DMA Controller can be used in association with the AES to perform an encryption/decryption of a buffer without any action by software during processing.

The SMOD field in the AES_MR must be configured to 0x2 and the DMA must be configured with non-incremental addresses.

The start address of any transfer descriptor must be configured with the address of AES_IDATAR0.

The DMA chunk size configuration depends on the AES mode of operation and is listed in Table 55-3 “DMA Data Transfer Type for the Different Operation Modes”.

When writing data to AES with a first DMA channel, data are first fetched from a memory buffer (source data). It is recommended to configure the size of source data to “words” even for CFB modes. On the contrary, the destination data size depends on the mode of operation. When reading data from the AES with the second DMA channel, the source data is the data read from AES and data destination is the memory buffer. In this case, the source data size depends on the AES mode of operation and is listed in Table 55-3.

Table 55-3. DMA Data Transfer Type for the Different Operation Modes

Operation Mode	Chunk Size	Destination/Source Data Transfer Type
ECB	4	Word
CBC	4	Word
OFB	4	Word
CFB 128-bit	4	Word
CFB 64-bit	1	Word
CFB 32-bit	1	Word
CFB 16-bit	1	Half-word
CFB 8-bit	1	Byte
CTR	4	Word
GCM	4	Word

55.4.5 Last Output Data Mode

This mode is used to generate cryptographic checksums on data (MAC) by means of cipher block chaining encryption algorithm (CBC-MAC algorithm for example).

Downloaded from Arrow.com.

After each end of encryption/decryption, the output data are available either on the AES_ODATARx for Manual and Auto mode or at the address specified in the receive buffer pointer for DMA mode (see Table 55-4 “Last Output Data Mode Behavior versus Start Modes”).

The Last Output Data (LOD) bit in the AES_MR allows retrieval of only the last data of several encryption/decryption processes.

Therefore, there is no need to define a read buffer in DMA mode.

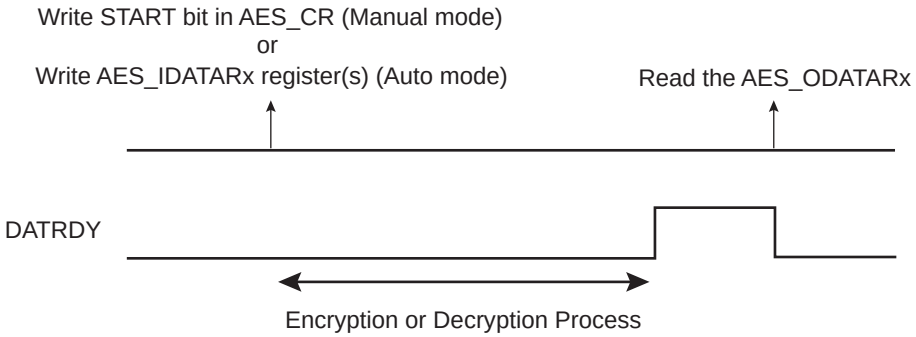
This data are only available on the AES_ODATARx.

55.4.5.1 Manual and Auto Modes

If AES_MR.LOD = 0

The DATRDY flag is cleared when at least one of the AES_ODATARx is read (see Figure 55-1).

Figure 55-1. Manual and Auto Modes with AES_MR.LOD = 0



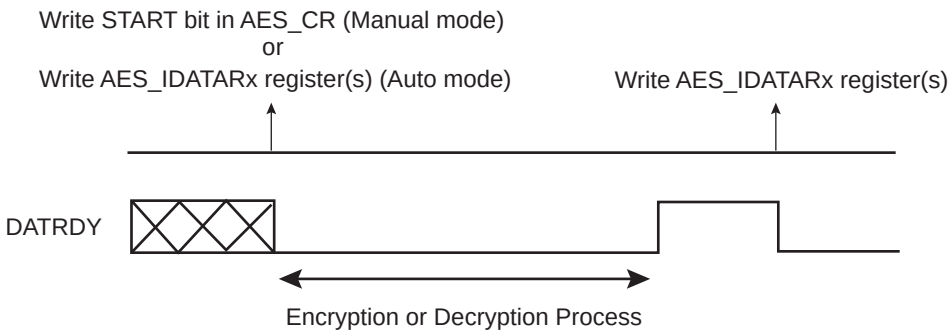
If the user does not want to read the AES_ODATARx between each encryption/decryption, the DATRDY flag will not be cleared. If the DATRDY flag is not cleared, the user cannot know the end of the following encryptions/decryptions.

If AES_MR.LOD = 1

This mode is optimized to process AES CPC-MAC operating mode.

The DATRDY flag is cleared when at least one AES_IDATAR is written (see Figure 55-2). No more AES_ODATAR reads are necessary between consecutive encryptions/decryptions.

Figure 55-2. Manual and Auto Modes with AES_MR.LOD = 1



Downloaded from Arrow.com.

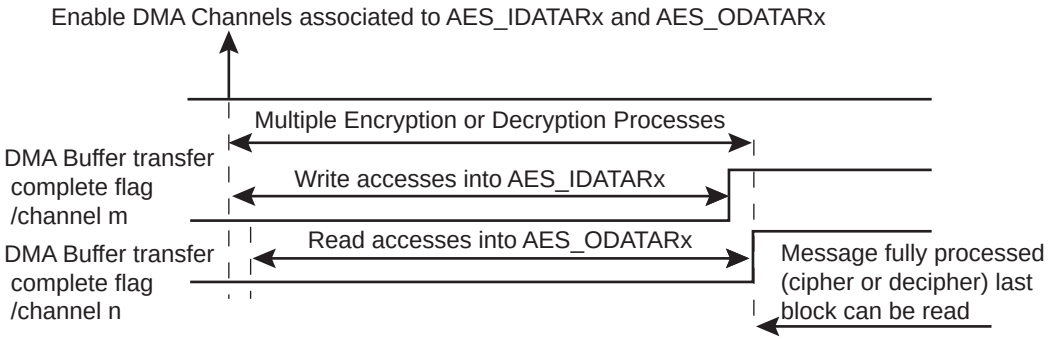
55.4.5.2 DMA Mode

If AES_MR.LOD = 0

This mode may be used for all AES operating modes except CBC-MAC where AES_MR.LOD = 1 mode is recommended.

The end of the encryption/decryption is indicated by the end of DMA transfer associated to AES_ODATARx (see Figure 55-3). Two DMA channels are required: one for writing message blocks to AES_IDATARx and one to obtain the result from AES_ODATARx.

Figure 55-3. DMA Transfer with AES_MR.LOD = 0



If AES_MR.LOD = 1

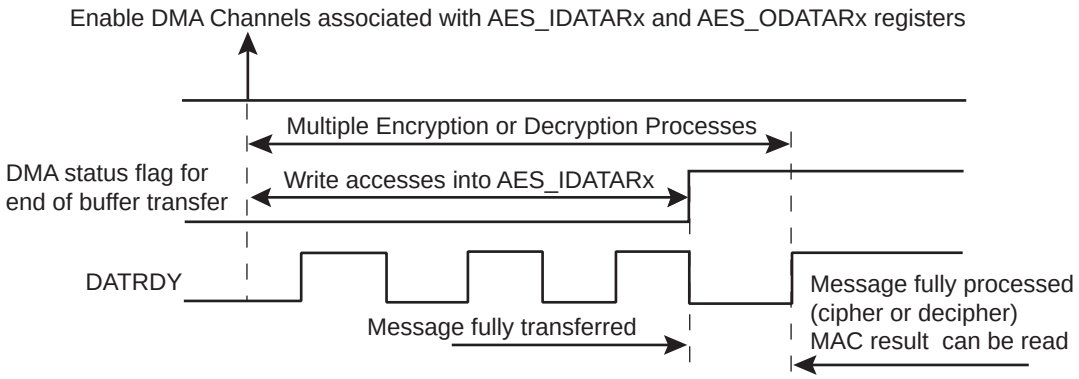
This mode is optimized to process AES CBC-MAC operating mode.

The user must first wait for the DMA buffer transfer complete flag, then for the flag DATRDY to rise to ensure that the encryption/decryption is completed (see Figure 55-4).

In this case, no receive buffers are required.

The output data are only available on the AES_ODATARx.

Figure 55-4. DMA Transfer with AES_MR.LOD = 1



Downloaded from Arrow.com.

Table 55-4 summarizes the different cases.

Table 55-4. Last Output Data Mode Behavior versus Start Modes

Sequence	Manual and Auto Modes		DMA Transfer	
	AES_MR.LOD = 0	AES_MR.LOD = 1	AES_MR.LOD = 0	AES_MR.LOD = 1
DATRDY Flag Clearing Condition(1)	At least one AES_ODATAR must be read	At least one AES_IDATAR must be written	Not used	Managed by the DMA
End of Encryption/Decryption Notification	DATRDY	DATRDY	2 DMA Buffer transfer complete flags (channel m and channel n)	DMA buffer transfer complete flag, then AES DATRDY flag
Encrypted/Decrypted Data Result Location	In the AES_ODATARx	In the AES_ODATARx	At the address specified in the Channel Buffer Transfer Descriptor	In the AES_ODATARx

Note: 1. Depending on the mode, there are other ways of clearing the DATRDY flag. See Section 55.5.6 “AES Interrupt Status Register”.

Warning: In DMA mode, reading the AES_ODATARx before the last data transfer may lead to unpredictable results.

Downloaded from Arrow.com.

55.4.6 Galois/Counter Mode (GCM)

55.4.6.1 Description

GCM comprises the AES engine in CTR mode along with a universal hash function (GHASH engine) that is defined over a binary Galois field to produce a message authentication tag (the AES CTR engine and the GHASH engine are depicted in Figure 55-5).

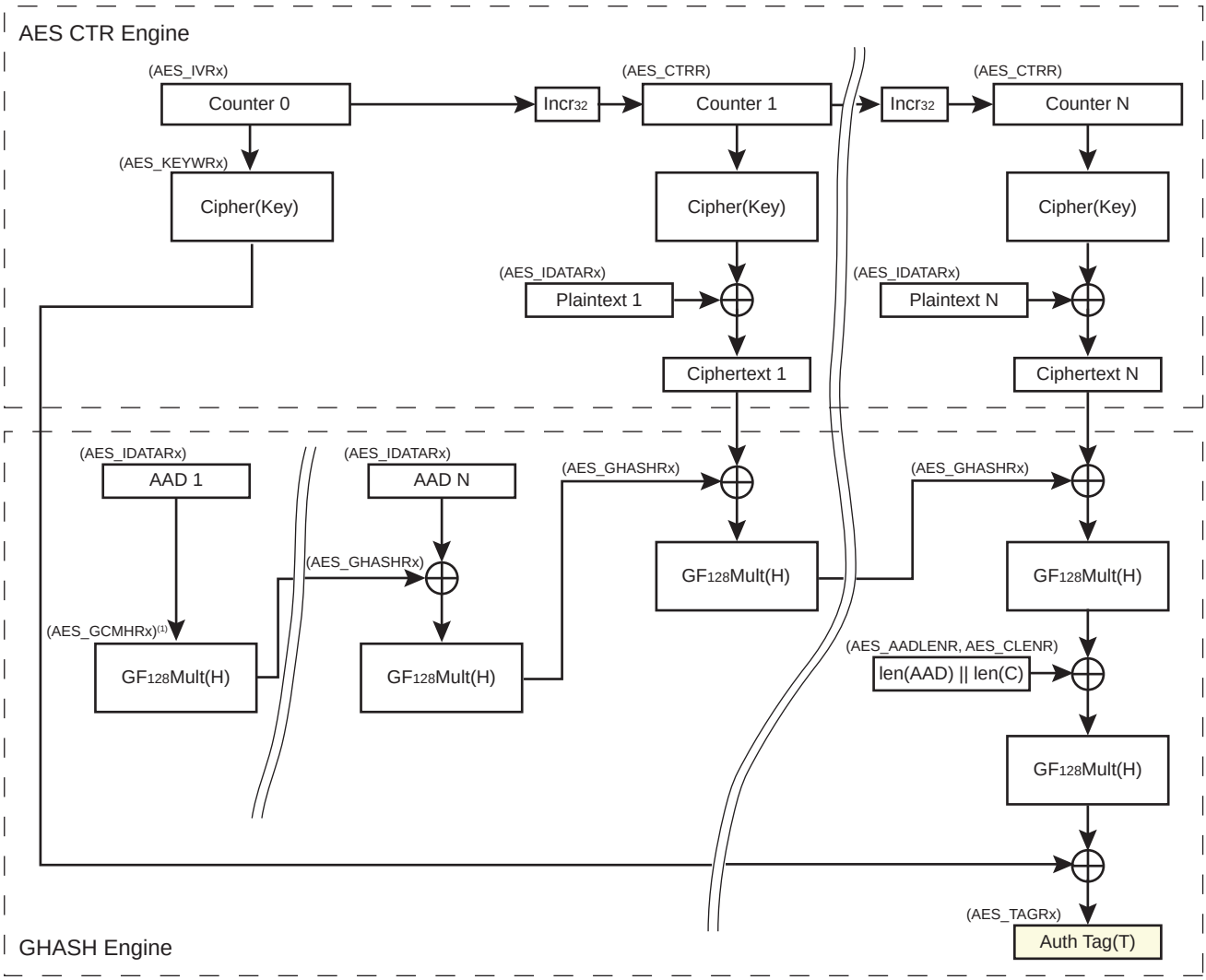
The GHASH engine processes data packets after the AES operation. GCM provides assurance of the confidentiality of data through the AES Counter mode of operation for encryption. Authenticity of the confidential data is assured through the GHASH engine. GCM can also provide assurance of data that is not encrypted. Refer to the *NIST Special Publication 800-38D* for more complete information.

GCM can be used with or without the DMA master. Messages may be processed as a single complete packet of data or they may be broken into multiple packets of data over time.

GCM processing is computed on 128-bit input data fields. There is no support for unaligned data. The AES key length can be whatever length is supported by the AES module.

The recommended programming procedure when using DMA is described in Section 55.4.6.3.

Figure 55-5. GCM Block Diagram



Notes: 1. Optional

Downloaded from Arrow.com.

55.4.6.2 Key Writing and Automatic Hash Subkey Calculation

Whenever a new key (AES_KEYWRx) is written to the hardware, two automatic actions are processed:

- GCM Hash Subkey *H* generation—The GCM hash subkey (*H*) is automatically generated. The GCM hash subkey generation must be complete before doing any other action. The DATRDY bit of the AES_ISR indicates when the subkey generation is complete (with interrupt if configured). The GCM hash subkey calculation is processed with the formula *H* = CIPHER(Key, <128 bits to zero>. The generated GCM *H* value is then available in the AES_GCMHRx. If the application software requires a specific hash subkey, the automatically generated H value can be overwritten in the AES_GCMHRx.
  The AES_GCMHRx can be written after the end of the hash subkey generation (see AES_ISR.DATRDY) and prior to starting the input data feed.
- AES_GHASHRx Clear—The AES_GHASHRx are automatically cleared. If a hash initial value is needed for the GHASH, it must be written to the AES_GHASHRx
  - after a write to AES_KEYWRx, if any
  - before starting the input data feed

55.4.6.3 GCM Processing

GCM processing comprises three phases:

1. Processing the Additional Authenticated Data (*AAD*), hash computation only.
2. Processing the Ciphertext (*C*), hash computation + ciphering/deciphering.
3. Generating the Tag using length of *AAD*, length of *C* and J_0 (see NIST documentation for details).

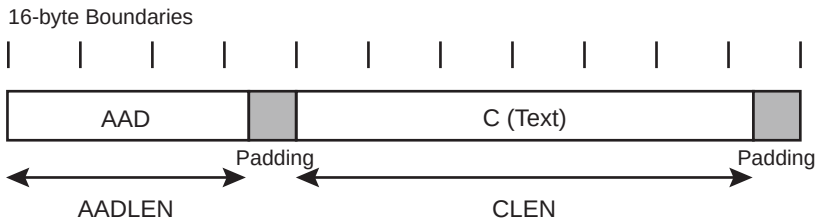
The Tag generation can be done either automatically, after the end of *AAD*/*C* processing if TAG_EN bit is set in the AES_MR or done manually, using the GHASH field in AES_GHASHRx (See subsections "Processing a Complete Message with Tag Generation" and "Manual GCM Tag Generation" below for details).

Processing a Complete Message with Tag Generation

Use this procedure only if J_0 four LSB bytes ≠ 0xFFFFFFFF.

NOTE: In the case where J_0 four LSB bytes = 0xFFFFFFFF or if the value is unknown, use the procedure described in "Processing a Complete Message without Tag Generation" followed by the procedure in "Manual GCM Tag Generation".

Figure 55-6. Full Message Alignment



To process a complete message with Tag generation, the sequence is as follows:

1. In AES_MR set OPMOD to GCM and GTAGEN to '1' (configuration as usual for the rest).
2. Set KEYW in AES_KEYWRx and wait until DATRDY bit of AES_ISR is set (GCM hash subkey generation complete); use interrupt if needed. See Section 55.4.6.2 "Key Writing and Automatic Hash Subkey Calculation" for details.
3. Calculate the J_0 value as described in NIST documentation $J_0 = IV \,||\, 0^{31} \,||\, 1$ when len(*IV*) = 96 and $J_0 = \text{GHASH}_H(IV \,||\, 0^{s+64} \,||\, [\text{len}(IV)]_{64})$ if len(*IV*) ≠ 96. See "Processing a Message with only AAD (GHASHH)" for J_0 generation.
4. Set IV in AES_IVRx with inc32(J_0) (J_0 + 1 on 32 bits).
5. Set AADLEN field in AES_AADLENR and CLEN field in AES_CLENR.

Downloaded from Arrow.com.

6. Fill the IDATA field of AES_IDATARx with the message to process according to the SMOD configuration used. If Manual Mode or Auto Mode is used, the DATRDY bit indicates when the data have been processed (however, no output data are generated when processing *AAD*).
7. Wait for TAGRDY to be set (use interrupt if needed), then read the TAG field of AES_TAGRx to obtain the authentication tag of the message.

Processing a Complete Message without Tag Generation

Processing a message without generating the Tag can be used to customize the Tag generation, or to process a fragmented message. To manually generate the GCM Tag, refer to "Manual GCM Tag Generation".

To process a complete message without Tag generation, the sequence is as follows:

1. In AES_MR set OPMOD to GCM and GTAGEN to '0' (configuration as usual for the rest).
2. Set KEYW in AES_KEYWRx and wait until DATRDY bit of AES_ISR is set (GCM hash subkey generation complete); use interrupt if needed. After the GCM hash subkey generation is complete the GCM hash subkey can be read or overwritten with specific value in the AES_GCMHRx (see Section 55.4.6.2 "Key Writing and Automatic Hash Subkey Calculation" for details).
3. Calculate the J_0 value as described in NIST documentation $J_0 = IV \,||\, 0^{31} \,||\, 1$ when len(IV) = 96 and $J_0 = \text{GHASH}_H(IV \,||\, 0^{s+64} \,||\, [\text{len}(IV)]_{64})$ if len(IV) ≠ 96. See "Processing a Message with only AAD (GHASHH)" for J_0 generation example when len(IV) ≠ 96.
4. Set IV in AES_IVRx with inc32(J_0) (J_0 + 1 on 32 bits).
5. Set AADLEN field in AES_AADLENR and CLEN field in AES_CLENR.
6. Fill the IDATA field of AES_IDATARx with the message to process according to the SMOD configuration used. If Manual Mode or Auto Mode is used, the DATRDY bit indicates when the data have been processed (however, no output data are generated when processing *AAD*).
7. Make sure the last output data have been read if CLEN ≠ 0 (or wait for DATRDY), then read the GHASH field of AES_GHASHRx to obtain the hash value after the last processed data.

Processing a Fragmented Message without Tag Generation

If needed, a message can be processed by fragments, in such case automatic GCM Tag generation is not supported.

To process a message by fragments, the sequence is as follows:

- First fragment:

1. In AES_MR set OPMOD to GCM and GTAGEN to '0' (configuration as usual for the rest).
2. Set KEYW in AES_KEYWRx and wait for DATRDY bit of AES_ISR to be set (GCM hash subkey generation complete); use interrupt if needed. After the GCM hash subkey generation is complete the GCM hash subkey can be read or overwritten with specific value in the AES_GCMHRx (see Section 55.4.6.2 "Key Writing and Automatic Hash Subkey Calculation" for details).
3. Calculate the J_0 value as described in NIST documentation $J_0 = IV \,||\, 0^{31} \,||\, 1$ when len(IV) = 96 and $J_0 = \text{GHASH}_H(IV \,||\, 0^{s+64} \,||\, [\text{len}(IV)]_{64})$ if len(IV) ≠ 96. See "Processing a Message with only AAD (GHASHH)" for J_0 generation example when len(IV) ≠ 96.
4. Set IV in AES_IVRx with inc32(J_0) (J_0 + 1 on 32 bits).
5. Set AADLEN field in AES_AADLENR and CLEN field in AES_CLENR according to the length of the first fragment, or set the fields with the full message length, both configurations work.
6. Fill the IDATA field of AES_IDATARx with the first fragment of the message to process (aligned on 16-byte boundary) according to the SMOD configuration used. If Manual Mode or Auto Mode is used the DATRDY bit indicates when the data have been processed (however, no output data are generated when processing *AAD*).
7. Make sure the last output data have been read if the fragment ends in C phase (or wait for DATRDY if the fragment ends in *AAD* phase), then read the GHASH field of AES_GHASHRx to obtain the value of the hash

Downloaded from Arrow.com.

after the last processed data and finally read the CTR field of the AES_CTR to obtain the value of the CTR encryption counter (not needed when the fragment ends in *AAD* phase).

- Next fragment (or last fragment):

1. In AES_MR set OPMOD to GCM and GTAGEN to '0' (configuration as usual for the rest).
2. Set KEYW in AES_KEYWRx and wait until DATRDY bit of AES_ISR is set (GCM hash subkey generation complete); use interrupt if needed. After the GCM hash subkey generation is complete the GCM hash subkey can be read or overwritten with specific value in the AES_GCMHRx (see Section 55.4.6.2 "Key Writing and Automatic Hash Subkey Calculation" for details).
3. Set IV in AES_IVRx with:
   - If the first block of the fragment is a block of Additional Authenticated data, set IV in AES_IVRx with the J0 initial value
   - If the first block of the fragment is a block of Plaintext data, set IV in AES_IVRx with a value constructed as follows: 'LSB96(J0) || CTR' value, (96 bit LSB of J0 concatenated with saved CTR value from previous fragment).
4. Set AADLEN field in AES_AADLENR and CLEN field in AES_CLENR according to the length of the current fragment, or set the fields with the remaining message length, both configurations work.
5. Fill the GHASH field of AES_GHASHRx with the value stored after the previous fragment.
6. Fill the IDATA field of AES_IDATARx with the current fragment of the message to process (aligned on 16 byte boundary) according to the SMOD configuration used. If Manual Mode or Auto Mode is used, the DATRDY bit indicates when the data have been processed (however, no output data are generated when processing *AAD*).
7. Make sure the last output data have been read if the fragment ends in C phase (or wait for DATRDY if the fragment ends in *AAD* phase), then read the GHASH field of AES_GHASHRx to obtain the value of the hash after the last processed data and finally read the CTR field of the AES_CTR to obtain the value of the CTR encryption counter (not needed when the fragment ends in *AAD* phase).

Note: Step 1 and 2 are required only if the value of the concerned registers has been modified.

Once the last fragment has been processed, the GHASH value will allow manual generation of the GCM tag (see "Manual GCM Tag Generation" for details).

Manual GCM Tag Generation

This section describes the last steps of the GCM Tag generation.

The Manual GCM Tag Generation is used to complete the GCM Tag Generation when the message has been processed without Tag Generation.

Note: The Message Processing without Tag Generation must be finished before processing the Manual GCM Tag Generation.

To generate a GCM Tag manually, the sequence is as follows:

Processing $S = \mathrm{GHASH}_H (AAD \,||\, 0^v \,||\, C \,||\, 0^u \,||\, [\mathrm{len}(AAD)]_{64} \,||\, [\mathrm{len}(C)]_{64})$:

1. In AES_MR set OPMOD to GCM and GTAGEN to '0' (configuration as usual for the rest).
2. Set KEYW in AES_KEYWRx and wait for DATRDY bit of AES_ISR to be set (GCM hash subkey generation complete); use interrupt if needed. After the GCM hash subkey generation is complete the GCM hash subkey can be read or overwritten with specific value in the AES_GCMHRx (see Section 55.4.6.2 "Key Writing and Automatic Hash Subkey Calculation" for details).
3. Set AADLEN field to 0x10 (16 bytes) in AES_AADLENR and CLEN field to '0' in AES_CLENR. This will allow running a single GHASH_H on a 16-byte input data (see Figure 55-7).
4. Fill the GHASH field of AES_GHASHRx with the state of the GHASH field stored at the end of the message processing.
5. Fill the IDATA field of AES_IDATARx according to the SMOD configuration used with '$\mathrm{len}(AAD)_{64} \,||\, \mathrm{len}(C)_{64}$' value as described in the NIST documentation and wait for DATRDY to be set; use interrupt if needed.

Downloaded from Arrow.com.

6. Read the GHASH field of AES_GHASHRx to obtain the current value of the hash.

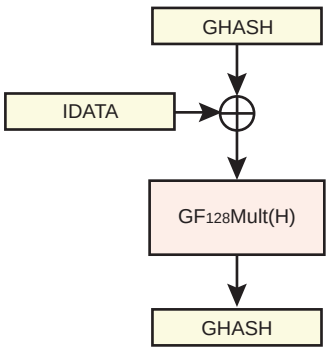
Processing $T = GCTR_K(J_0, S)$:

7. In AES_MR set OPMOD to CTR (configuration as usual for the rest).
8. Set the IV field in AES_IVRx with 'J_0' value.
9. Fill the IDATA field of AES_IDATARx with the GHASH value read at step 6 and wait for DATRDY to be set (use interrupt if needed).
10. Read the ODATA field of AES_ODATARx to obtain the GCM Tag value.

Note: Step 4 is optional if the GHASH field is to be filled with value '0' (0 length packet for instance).

Processing a Message with only AAD ($GHASH_H$)

Figure 55-7. Single $GHASH_H$ Block Diagram (AADLEN ≤ 0x10 and CLEN = 0)



It is possible to process a message with only *AAD* setting the CLEN field to '0' in the AES_CLENR, this can be used for J_0 generation when len(*IV*) ≠ 96 for instance.

Example: Processing J_0 when len(*IV*) ≠ 96

To process $J_0 = GHASH_H(IV \,||\, 0^{s+64} \,||\, [len(IV)]_{64})$, the sequence is as follows:

1. In AES_MR set OPMOD to GCM and GTAGEN to '0' (configuration as usual for the rest).
2. Set KEYW in AES_KEYWRx and wait until DATRDY bit of AES_ISR is set (GCM hash subkey generation complete); use interrupt if needed. After the GCM hash subkey generation is complete the GCM hash subkey can be read or overwritten with specific value in the AES_GCMHRx (see Section 55.4.6.2 "Key Writing and Automatic Hash Subkey Calculation" for details).
3. Set AADLEN field with '$len(IV \,||\, 0^{s+64} \,||\, [len(IV)]_{64})$' in AES_AADLENR and CLEN field to '0' in AES_CLENR. This will allow running a $GHASH_H$ only.
4. Fill the IDATA field of AES_IDATARx with the message to process ($IV \,||\, 0^{s+64} \,||\, [len(IV)]_{64}$) according to the SMOD configuration used. If Manual Mode or Auto Mode is used, the DATRDY bit indicates when a $GHASH_H$ step is over (use interrupt if needed).
5. Read the GHASH field of AES_GHASHRx to obtain the J_0 value.

Note: The GHASH value can be overwritten at any time by writing the GHASH field value of AES_GHASHRx, used to perform a $GHASH_H$ with an initial value for GHASH (write GHASH field between step 3 and step 4 in this case).

Processing a Single GF_{128} Multiplication

The AES can also be used to process a single multiplication in the Galois field on 128 bits (GF_{128}) using a single $GHASH_H$ with custom *H* value (see Figure 55-7).

To run a GF_{128} multiplication (A x B), the sequence is as follows:

1. In AES_MR set OPMOD to GCM and GTAGEN to '0' (configuration as usual for the rest).
2. Set AADLEN field with 0x10 (16 bytes) in AES_AADLENR and CLEN field to '0' in AES_CLENR. This will allow running a single $GHASH_H$.
3. Fill the H field of the AES_GCMHRx with B value.

Downloaded from Arrow.com.

4. Fill the IDATA field of AES_IDATARx with the A value according to the SMOD configuration used. If Manual Mode or Auto Mode is used, the DATRDY bit indicates when a GHASH$_H$ computation is over (use interrupt if needed).
5. Read the GHASH field of AES_GHASHRx to obtain the result.

Note: The GHASH field of AES_GHASHRx can be initialized with a value C between step 3 and step 4 to run a ((A XOR C) x B) GF_{128} multiplication.

55.4.7 Security Features

55.4.7.1 Unspecified Register Access Detection

When an unspecified register access occurs, the URAD flag in the AES_ISR is raised. Its source is then reported in the Unspecified Register Access Type (URAT) field. Only the last unspecified register access is available through the URAT field.

Several kinds of unspecified register accesses can occur:

- Input Data Register written during the data processing when SMOD = IDATAR0_START
- Output Data Register read during data processing
- Mode Register written during data processing
- Output Data Register read during sub-keys generation
- Mode Register written during sub-keys generation
- Write-only register read access

The URAD bit and the URAT field can only be reset by the SWRST bit in the AES_CR.

Downloaded from Arrow.com.

55.5 Advanced Encryption Standard (AES) User Interface

Table 55-5. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	AES_CR	Write-only	–
0x04	Mode Register	AES_MR	Read/Write	0x0
0x08–0x0C	Reserved	–	–	–
0x10	Interrupt Enable Register	AES_IER	Write-only	–
0x14	Interrupt Disable Register	AES_IDR	Write-only	–
0x18	Interrupt Mask Register	AES_IMR	Read-only	0x0
0x1C	Interrupt Status Register	AES_ISR	Read-only	0x0
0x20	Key Word Register 0	AES_KEYWR0	Write-only	–
0x24	Key Word Register 1	AES_KEYWR1	Write-only	–
0x28	Key Word Register 2	AES_KEYWR2	Write-only	–
0x2C	Key Word Register 3	AES_KEYWR3	Write-only	–
0x30	Key Word Register 4	AES_KEYWR4	Write-only	–
0x34	Key Word Register 5	AES_KEYWR5	Write-only	–
0x38	Key Word Register 6	AES_KEYWR6	Write-only	–
0x3C	Key Word Register 7	AES_KEYWR7	Write-only	–
0x40	Input Data Register 0	AES_IDATAR0	Write-only	–
0x44	Input Data Register 1	AES_IDATAR1	Write-only	–
0x48	Input Data Register 2	AES_IDATAR2	Write-only	–
0x4C	Input Data Register 3	AES_IDATAR3	Write-only	–
0x50	Output Data Register 0	AES_ODATAR0	Read-only	0x0
0x54	Output Data Register 1	AES_ODATAR1	Read-only	0x0
0x58	Output Data Register 2	AES_ODATAR2	Read-only	0x0
0x5C	Output Data Register 3	AES_ODATAR3	Read-only	0x0
0x60	Initialization Vector Register 0	AES_IVR0	Write-only	–
0x64	Initialization Vector Register 1	AES_IVR1	Write-only	–
0x68	Initialization Vector Register 2	AES_IVR2	Write-only	–
0x6C	Initialization Vector Register 3	AES_IVR3	Write-only	–
0x70	Additional Authenticated Data Length Register	AES_AADLENR	Read/Write	–
0x74	Plaintext/Ciphertext Length Register	AES_CLENR	Read/Write	–
0x78	GCM Intermediate Hash Word Register 0	AES_GHASHR0	Read/Write	–
0x7C	GCM Intermediate Hash Word Register 1	AES_GHASHR1	Read/Write	–
0x80	GCM Intermediate Hash Word Register 2	AES_GHASHR2	Read/Write	–
0x84	GCM Intermediate Hash Word Register 3	AES_GHASHR3	Read/Write	–
0x88	GCM Authentication Tag Word Register 0	AES_TAGR0	Read-only	–
0x8C	GCM Authentication Tag Word Register 1	AES_TAGR1	Read-only	–

Downloaded from Arrow.com.

Table 55-5. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x90	GCM Authentication Tag Word Register 2	AES_TAGR2	Read-only	_
0x94	GCM Authentication Tag Word Register 3	AES_TAGR3	Read-only	_
0x98	GCM Encryption Counter Value Register	AES_CTRR	Read-only	_
0x9C	GCM H Word Register 0	AES_GCMHR0	Read/Write	_
0xA0	GCM H Word Register 1	AES_GCMHR1	Read/Write	_
0xA4	GCM H Word Register 2	AES_GCMHR2	Read/Write	_
0xA8	GCM H Word Register 3	AES_GCMHR3	Read/Write	_
0xAC	Reserved	–	_	_
0xB0–0xFC	Reserved	–	_	_

Downloaded from Arrow.com.

55.5.1 AES Control Register

Name: AES_CR

Address: 0x4006C000

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	SWRST
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	START

- **START: Start Processing**

0: No effect.

1: Starts manual encryption/decryption process.

- **SWRST: Software Reset**

0: No effect.

1: Resets the AES. A software-triggered hardware reset of the AES interface is performed.

Downloaded from Arrow.com.

55.5.2 AES Mode Register

Name: AES_MR

Address: 0x4006C004

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
CKEY				–	CFBS		
15	14	13	12	11	10	9	8
LOD	OPMOD			KEYSIZE		SMOD	
7	6	5	4	3	2	1	0
PROCDLY				DUALBUFF	–	GTAGEN	CIPHER

- **CIPHER: Processing Mode**

0: Decrypts data.

1: Encrypts data.

- **GTAGEN: GCM Automatic Tag Generation Enable**

0: Automatic GCM Tag generation disabled.

1: Automatic GCM Tag generation enabled.

- **DUALBUFF: Dual Input Buffer**

Value	Name	Description
0	INACTIVE	AES_IDATARx cannot be written during processing of previous block.
1	ACTIVE	AES_IDATARx can be written during processing of previous block when SMOD = 2. It speeds up the overall runtime of large files.

- **PROCDLY: Processing Delay**

Processing Time = N × (PROCDLY + 1)

where

N = 10 when KEYSIZE = 0

N = 12 when KEYSIZE = 1

N = 14 when KEYSIZE = 2

The processing time represents the number of clock cycles that the AES needs in order to perform one encryption/decryption.

Note: The best performance is achieved with PROCDLY equal to 0.

- **SMOD: Start Mode**

Value	Name	Description
0	MANUAL_START	Manual Mode
1	AUTO_START	Auto Mode
2	IDATAR0_START	AES_IDATAR0 access only Auto Mode (DMA)

Downloaded from Arrow.com.

Values which are not listed in the table must be considered as “reserved”.

If a DMA transfer is used, configure SMOD to 0x2. Refer to Section 55.4.4.3 “DMA Mode” for more details.

- **KEYSIZE: Key Size**

Value	Name	Description
0	AES128	AES Key Size is 128 bits
1	AES192	AES Key Size is 192 bits
2	AES256	AES Key Size is 256 bits

Values which are not listed in the table must be considered as “reserved”.

- **OPMOD: Operation Mode**

Value	Name	Description
0	ECB	ECB: Electronic Code Book mode
1	CBC	CBC: Cipher Block Chaining mode
2	OFB	OFB: Output Feedback mode
3	CFB	CFB: Cipher Feedback mode
4	CTR	CTR: Counter mode (16-bit internal counter)
5	GCM	GCM: Galois/Counter mode

Values which are not listed in the table must be considered as “reserved”.

For CBC-MAC operating mode, set OPMOD to CBC and LOD to 1.

- **LOD: Last Output Data Mode**

0: No effect.

After each end of encryption/decryption, the output data are available either on the output data registers (Manual and Auto modes) or at the address specified in the Channel Buffer Transfer Descriptor for DMA mode.

In Manual and Auto modes, the DATRDY flag is cleared when at least one of the Output Data registers is read.

1: The DATRDY flag is cleared when at least one of the Input Data Registers is written.

No more Output Data Register reads is necessary between consecutive encryptions/decryptions (see Section 55.4.5 “Last Output Data Mode”).

Warning: In DMA mode, reading to the Output Data registers before the last data encryption/decryption process may lead to unpredictable results.

- **CFBS: Cipher Feedback Data Size**

Value	Name	Description
0	SIZE_128BIT	128-bit
1	SIZE_64BIT	64-bit
2	SIZE_32BIT	32-bit
3	SIZE_16BIT	16-bit
4	SIZE_8BIT	8-bit

Values which are not listed in table must be considered as “reserved”.

Downloaded from Arrow.com.

- **CKEY: Key**

Value	Name	Description
0xE	PASSWD	This field must be written with 0xE the first time the AES_MR is programmed. For subsequent programming of the AES_MR, any value can be written, including that of 0xE. Always reads as 0.

Note:

55.5.3 AES Interrupt Enable Register

Name: AES_IER

Address: 0x4006C010

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	TAGRDY
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

- **DATRDY: Data Ready Interrupt Enable**
- **URAD: Unspecified Register Access Detection Interrupt Enable**
- **TAGRDY: GCM Tag Ready Interrupt Enable**

Downloaded from Arrow.com.

55.5.4 AES Interrupt Disable Register

Name: AES_IDR

Address: 0x4006C014

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	TAGRDY
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

- **DATRDY: Data Ready Interrupt Disable**
- **URAD: Unspecified Register Access Detection Interrupt Disable**
- **TAGRDY: GCM Tag Ready Interrupt Disable**

Downloaded from Arrow.com.

55.5.5 AES Interrupt Mask Register

Name: AES_IMR

Address: 0x4006C018

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	TAGRDY
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

- **DATRDY: Data Ready Interrupt Mask**
- **URAD: Unspecified Register Access Detection Interrupt Mask**
- **TAGRDY: GCM Tag Ready Interrupt Mask**

Downloaded from Arrow.com.

55.5.6 AES Interrupt Status Register

Name: AES_ISR

Address: 0x4006C01C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	TAGRDY
15	14	13	12	11	10	9	8
URAT				–	–	–	URAD
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DATRDY

- **DATRDY: Data Ready (cleared by setting bit START or bit SWRST in AES_CR or by reading AES_ODATARx)**

0: Output data not valid.

1: Encryption or decryption process is completed.

Note: If AES_MR.LOD = 1: In Manual and Auto mode, the DATRDY flag can also be cleared by writing at least one AES_IDATARx.

- **URAD: Unspecified Register Access Detection Status (cleared by writing SWRST in AES_CR)**

0: No unspecified register access has been detected since the last SWRST.

1: At least one unspecified register access has been detected since the last SWRST.

- **URAT: Unspecified Register Access (cleared by writing SWRST in AES_CR)**

Value	Name	Description
0	IDR_WR_PROCESSING	Input Data Register written during the data processing when SMOD = 0x2 mode.
1	ODR_RD_PROCESSING	Output Data Register read during the data processing.
2	MR_WR_PROCESSING	Mode Register written during the data processing.
3	ODR_RD_SUBKGEN	Output Data Register read during the sub-keys generation.
4	MR_WR_SUBKGEN	Mode Register written during the sub-keys generation.
5	WOR_RD_ACCESS	Write-only register read access.

Only the last Unspecified Register Access Type is available through the URAT field.

- **TAGRDY: GCM Tag Ready**

0: GCM Tag is not valid.

1: GCM Tag generation is complete (cleared by reading GCM Tag, starting another processing or when writing a new key).

Downloaded from Arrow.com.

55.5.7 AES Key Word Register x

Name: AES_KEYWRx [x=0..7]

Address: 0x4006C020

Access: Write-only

31	30	29	28	27	26	25	24
KEYW							

23	22	21	20	19	18	17	16
KEYW							

15	14	13	12	11	10	9	8
KEYW							

7	6	5	4	3	2	1	0
KEYW							

- **KEYW: Key Word**

The four/six/eight 32-bit Key Word Registers set the 128-bit/192-bit/256-bit cryptographic key used for AES encryption/decryption.

AES_KEYWR0 corresponds to the first word of the key and respectively AES_KEYWR3/AES_KEYWR5/AES_KEYWR7 to the last one.

Whenever a new key (AES_KEYWRx) is written to the hardware, two automatic actions are processed:

- GCM hash subkey generation
- AES_GHASHRx Clear

See Section 55.4.6.2 for details.

These registers are write-only to prevent the key from being read by another application.

Downloaded from Arrow.com.

55.5.8 AES Input Data Register x

Name: AES_IDATARx [x=0..3]

Address: 0x4006C040

Access: Write-only

31	30	29	28	27	26	25	24
IDATA							
23	22	21	20	19	18	17	16
IDATA							
15	14	13	12	11	10	9	8
IDATA							
7	6	5	4	3	2	1	0
IDATA							

- **IDATA: Input Data Word**

The four 32-bit Input Data registers set the 128-bit data block used for encryption/decryption.

AES_IDATAR0 corresponds to the first word of the data to be encrypted/decrypted, and AES_IDATAR3 to the last one.

These registers are write-only to prevent the input data from being read by another application.

Downloaded from Arrow.com.

55.5.9 AES Output Data Register x

Name: AES_ODATARx [x=0..3]

Address: 0x4006C050

Access: Read-only

31	30	29	28	27	26	25	24
ODATA							
23	22	21	20	19	18	17	16
ODATA							
15	14	13	12	11	10	9	8
ODATA							
7	6	5	4	3	2	1	0
ODATA							

- **ODATA: Output Data**

The four 32-bit Output Data registers contain the 128-bit data block that has been encrypted/decrypted.

AES_ODATAR0 corresponds to the first word, AES_ODATAR3 to the last one.

Downloaded from Arrow.com.

55.5.10 AES Initialization Vector Register x

Name: AES_IVRx [x=0..3]

Address: 0x4006C060

Access: Write-only

31	30	29	28	27	26	25	24
IV							
23	22	21	20	19	18	17	16
IV							
15	14	13	12	11	10	9	8
IV							
7	6	5	4	3	2	1	0
IV							

- **IV: Initialization Vector**

The four 32-bit Initialization Vector Registers set the 128-bit Initialization Vector data block that is used by some modes of operation as an additional initial input.

AES_IVR0 corresponds to the first word of the Initialization Vector, AES_IVR3 to the last one.

These registers are write-only to prevent the Initialization Vector from being read by another application.

For CBC, OFB and CFB modes, the IV input value corresponds to the initialization vector.

For CTR mode, the IV input value corresponds to the initial counter value.

Note: These registers are not used in ECB mode and must not be written.

Downloaded from Arrow.com.

55.5.11 AES Additional Authenticated Data Length Register

Name: AES_AADLENR

Address: 0x4006C070

Access: Read/Write

31	30	29	28	27	26	25	24
AADLEN							
23	22	21	20	19	18	17	16
AADLEN							
15	14	13	12	11	10	9	8
AADLEN							
7	6	5	4	3	2	1	0
AADLEN							

- **AADLEN: Additional Authenticated Data Length**

Length in bytes of the Additional Authenticated Data (*AAD*) that is to be processed.

Note: The maximum byte length of the *AAD* portion of a message is limited to the 32-bit counter length.

Downloaded from Arrow.com.

55.5.12 AES Plaintext/Ciphertext Length Register

Name: AES_CLENR

Address: 0x4006C074

Access: Read/Write

31	30	29	28	27	26	25	24
CLEN							

23	22	21	20	19	18	17	16
CLEN							

15	14	13	12	11	10	9	8
CLEN							

7	6	5	4	3	2	1	0
CLEN							

- **CLEN: Plaintext/Ciphertext Length**

Length in bytes of the plaintext/ciphertext (*C*) data that is to be processed.

Note: The maximum byte length of the C portion of a message is limited to the 32-bit counter length.

Downloaded from Arrow.com.

55.5.13 AES GCM Intermediate Hash Word Register x

Name: AES_GHASHRx [x=0..3]

Address: 0x4006C078

Access: Read/Write

31	30	29	28	27	26	25	24
GHASH							
23	22	21	20	19	18	17	16
GHASH							
15	14	13	12	11	10	9	8
GHASH							
7	6	5	4	3	2	1	0
GHASH							

- **GHASH: Intermediate GCM Hash Word x**

The four 32-bit Intermediate Hash Word registers expose the intermediate GHASH value. May be read to save the current GHASH value so processing can later be resumed, presumably on a later message fragment. Whenever a new key (AES_KEYWRx) is written to the hardware two automatic actions are processed:

- GCM hash subkey generation
- AES_GHASHRx Clear

See Section 55.4.6.2 for details.

If an application software specific hash initial value is needed for the GHASH it must be written to the AES_GHASHRx:

- after a write to AES_KEYWRx, if any
- before starting the input data feed

Downloaded from Arrow.com.

55.5.14 AES GCM Authentication Tag Word Register x

Name: AES_TAGRx [x=0..3]

Address: 0x4006C088

Access: Read-only

31	30	29	28	27	26	25	24
TAG							

23	22	21	20	19	18	17	16
TAG							

15	14	13	12	11	10	9	8
TAG							

7	6	5	4	3	2	1	0
TAG							

- **TAG: GCM Authentication Tag x**

The four 32-bit Tag registers contain the final 128-bit GCM Authentication tag (*T*) when GCM processing is complete. TAG0 corresponds to the first word, TAG3 to the last word.

Downloaded from Arrow.com.

55.5.15 AES GCM Encryption Counter Value Register

Name: AES_CTRR

Address: 0x4006C098

Access: Read-only

31	30	29	28	27	26	25	24
CTR							
23	22	21	20	19	18	17	16
CTR							
15	14	13	12	11	10	9	8
CTR							
7	6	5	4	3	2	1	0
CTR							

- **CTR: GCM Encryption Counter**

Reports the current value of the 32-bit GCM counter.

Downloaded from Arrow.com.

55.5.16 AES GCM H Word Register x

Name: AES_GCMHRx [x=0..3]

Address: 0x4006C09C

Access: Read/Write

31	30	29	28	27	26	25	24
H							

23	22	21	20	19	18	17	16
H							

15	14	13	12	11	10	9	8
H							

7	6	5	4	3	2	1	0
H							

- **H: GCM H Word x**

The four 32-bit H Word registers contain the 128-bit GCM hash subkey *H* value.

Whenever a new key (AES_KEYWRx) is written to the hardware two automatic actions are processed:

- GCM hash subkey *H* generation
- AES_GHASHRx Clear

If the application software requires a specific hash subkey, the automatically generated H value can be overwritten in the AES_GCMHRx (see Section 55.4.6.2 for details).

The choice of a GCM hash subkey *H* by a write in the AES_GCMHRx permits

- selection of the GCM hash subkey *H* for GHASH operations
- selection of one operand to process a single GF128 multiply

Downloaded from Arrow.com.

56. Electrical Characteristics

56.1 Absolute Maximum Ratings

Table 56-1. Absolute Maximum Ratings*

Storage Temperature. . . . . -60°C to + 150°C Voltage on Input Pins with Respect to Ground. . . . . -0.3V to + 4.0V Maximum Operating Voltage VDDPLL, VDDUTMIC, VDDCORE . . . . . 1.4V Maximum Operating Voltage VDDIO, VDDUTMII, VDDPLLUSB, VDDIN . . . . . 4.0V Total DC Output Current on all I/O lines 144-lead LQFP . . . . . 150 mA 144-ball LFBGA. . . . . 150 mA 144-ball UFBGA . . . . . 150 mA 100-lead LQFP . . . . . 150 mA 100-ball TFBGA. . . . . 150 mA 64-lead LQFP . . . . . 150 mA	*Notice: Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Downloaded from Arrow.com.

56.2 DC Characteristics

The following characteristics are applicable to the operating temperature range: T_A [-40°C : +105°C], unless otherwise specified.

Table 56-2. DC Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
T_A	Operating Temperature	–	-40	–	105	°C
T_J	Junction Temperature	–	-40	–	125	°C
V_{DDCORE}	DC Supply Core	–	1.08	1.2	1.32	V
	Allowable Voltage Ripple	rms value 10 kHz to 20 MHz	–	–	20	mV
	Slope	–	2.4	–	30	V/ms
V_{DDIO}	DC Supply I/Os, Backup	(1)	1.7	3.3	3.6	V
	Allowable Voltage Ripple	rms value 10 kHz to 10 MHz	–	–	30	mV
	Slope	–	1.9	–	30	V/ms
V_{DDIN}	DC Supply Voltage regulator, AFE, DAC, ACC	(1)	1.7	3.3	3.6	V
	Allowable Voltage Ripple	rms value 10 kHz to 20 MHz	–	–	20	mV
V_{DDPLL}	PLL A and Main Oscillator Supply	–	1.08	1.2	1.32	V
	Allowable Voltage Ripple	rms value 10 kHz to 10 MHz	–	–	20	mV
		rms value > 10 MHz	–	–	10	
$V_{DDUTMIC}$	DC Supply UDPHS and UHPHS UTMI+ Core	–	1.08	1.2	1.32	V
	Allowable Voltage Ripple	rms value 10 kHz to 10 MHz	–	–	10	mV
$V_{DDUTMII}$	DC Supply UDPHS and UHPHS UTMI+ Interface	–	3	3.3	3.6	V
	Allowable Voltage Ripple	rms value 10 kHz to 10 MHz	–	–	20	mV
$V_{DDPLLUSB}$	DC Supply UTMI PLL	–	3	3.3	3.6	V
	Allowable Voltage Ripple	rms value 10 kHz to 10 MHz	–	–	10	mV
V_{IL}	Low-level Input Voltage(2)	GPIO_MLB	-0.3	–	0.7	V
		GPIO_AD, GPIO_CLK	-0.3		MIN[0.8:0.3 x V_{DDIO}]	
		GPIO, CLOCK	-0.3	–	0.3 x V_{DDIO}	
V_{IH}	High-level Input Voltage(2)	GPIO_MLB	1.8	–	V_{DDIO} +0.3	V
		GPIO_AD, GPIO_CLK	MIN[2.0:0.7 x V_{DDIO}]		V_{DDIO} +0.3	
		GPIO, CLOCK	0.7 x V_{DDIO}	–	V_{DDIO} +0.3	
V_{OH}	High-level Output Voltage	GPIO_MLB, I_{OH} = 2.0 mA	V_{DDIO} - 0.4	–	–	V
		GPIO_AD, I_{OH} = 4.0 mA	V_{DDIO} - 0.4	–	–	
		GPIO_CLK, I_{OH} = 6.0 mA	V_{DDIO} - 0.4	–	–	
		GPIO, CLOCK, I_{OH} = 4.0mA	V_{DDIO} - 0.4	–	–	

Note:
1. VDDIO voltage must be equal to VDDIN voltage.
2. Current injection may lead to performance degradation or functional failures.

Downloaded from Arrow.com.

Table 56-3. DC Characteristics

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
V_{OL}	Low-level Output Voltage	GPIO_MLB, I_{OL}= -6.0 mA		–	–	0.4	V
		GPIO_AD, I_{OL} = -4.0 mA		–	–	0.4	
		GPIO_CLK, I_{OL}= -6.0 mA		–	–	0.4	
		GPIO, CLOCK, I_{OL} = -4.0 mA		–	–	0.4	
V_{hys}	Hysteresis Voltage	GPIO with Hysteresis mode enabled		150	–	–	mV
I_O	I_{OH} (or I_{SOURCE})	GPIO_MLB	V_{OH} = V_{DDIO} - 0.4	–	–	2	mA
		GPIO_CLK, low drive		–	–	6	
		GPIO_CLK, high drive		–	–	12	
		GPIO_AD, GPIO, low drive		–	–	4	
		GPIO_AD, GPIO, CLOCK, high drive		–	–	10	
I_O	I_{OL} (or I_{SINK})	GPIO_MLB	V_{OL} = 0.4V	–	–	-2	mA
		GPIO_CLK		–	–	-6	
		GPIO_AD, GPIO, CLOCK		–	–	-4	
I_{IL}	Low-level Input Current	Pull-up OFF		-1	–	1	µA
		Pull-up ON		10	–	50	
I_{IH}	High-level Input Current	Pull-down OFF		-1	–	1	µA
		Pull-down ON		10	–	50	
R_{PULLUP}	Pull-up Resistor	GPIO_MLB, GPIO_AD, GPIO_CLK		50	100	150	kOhm
		GPIO, CLOCK		70	100	140	
$R_{PULLDOWN}$	Pull-down Resistor	GPIO_MLB, GPIO_AD, GPIO_CLK		50	100	150	kOhm
		GPIO, CLOCK		70	100	140	
R_{SERIAL}	Serial Resistor	GPIO_MLB		–	9	–	
		GPIO_AD, GPIO_CLK		–	14	–	
		GPIO, CLOCK		–	26	–	Ohm

Downloaded from Arrow.com.

Table 56-4. 1.2V Voltage Regulator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{DDOUT}	DC Output Voltage	Normal mode, I_{LOAD} = 100 mA	1.2	1.23	1.26	V
		Standby mode	–	0	–	
I_{LOAD}	Maximum DC Output Current		–	–	150	mA
V_{LOAD}	Load Regulation	V_{DDIN} > 2V; I_{LOAD} = 10% to 90% max	–	20	40	mV
I_Q	Quiescent Current	Normal mode, I_{LOAD} = 0 mA	–	7.5	–	µA
		Normal mode, I_{LOAD} = max	–	500	–	µA
CD_{IN}	Input Decoupling Capacitor	(1)	–	4.7	–	µF
CD_{OUT}	Output Decoupling Capacitor	(2)	–	1	–	µF
		ESR	–	–	2	Ohm
t_{ON}	Turn-on Time	CD_{OUT}= 1 µF, V_{DDOUT} reaches DC output voltage	–	1	2.5	ms

Notes:
1. A 4.7 µF (±20%) or higher ceramic capacitor must be connected between VDDIN and the closest GND pin of the device. This large decoupling capacitor is mandatory to reduce startup current, improving transient response and noise rejection.
2. To ensure stability, an external 1 µF (±20%) output capacitor, CD_{OUT}, must be connected between VDDOUT and the closest GND pin of the device. Solid tantalum and multilayer ceramic capacitors are suitable as output capacitors. A 100 nF bypass capacitor between VDDOUT and the closest GND pin of the device helps decrease output noise and improves the load transient response.

Downloaded from Arrow.com.

Table 56-5. Core Power Supply Brownout Detector Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{T-}	Supply Falling Threshold(1)	–	0.97	1.0	1.04	V
V_{hys}	Hysteresis Voltage	–	–	25	50	mV
I_{DDON}	Current Consumption on VDDCORE	Brownout detector enabled(1)	–	0.5	2	µA
I_{DD33ON}	Current Consumption on VDDIO	Brownout detector enabled(1)	–	15	22	µA
t_{START}	Startup Time	From disabled state to enabled state	–	–	400	µs

Note: 1. The brownout detector is configured using the bit BODDIS in SUPC_MR.

Figure 56-1. Core Brownout Output Waveform

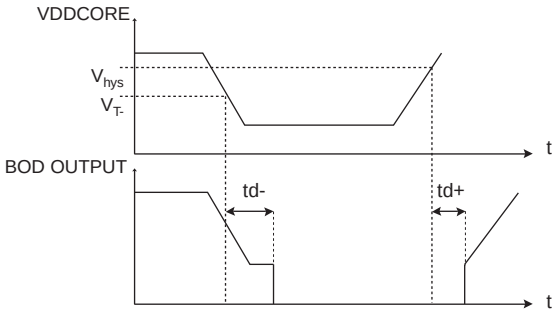
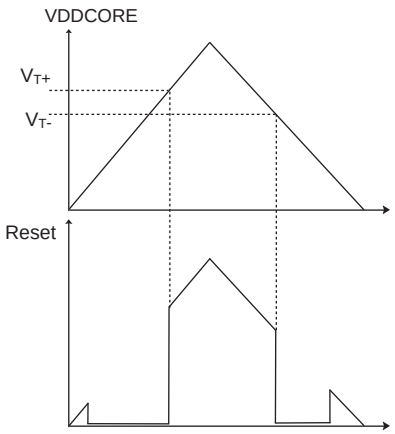


Table 56-6. VDDCORE Power-On Reset Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{T+}	Threshold Voltage Rising	–	0.79	0.95	1.07	V
V_{T-}	Threshold Voltage Falling	–	0.66	0.89	–	V
V_{hys}	Hysteresis Voltage	–	10	60	110	mV
t_{RES}	Reset Timeout Period	–	120	350	800	µs

Figure 56-2. VDDCORE Power-On Reset Characteristics



Downloaded from Arrow.com.

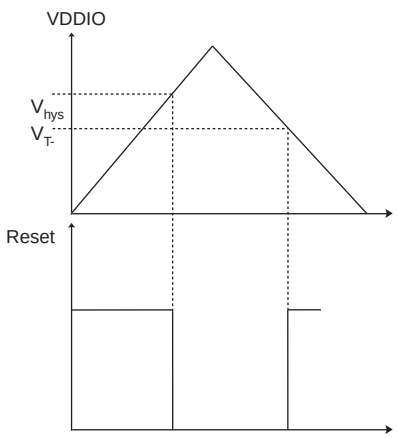
Table 56-7. VDDIO Supply Monitor

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{T-}	Supply Monitor Threshold	16 selectable steps See Threshold Selection table below	–	–	–	V
T_{ACC}	Threshold Accuracy		-4	–	4	%
V_{hys}	Hysteresis Voltage	–	–	38	45	mV
I_{DDON}	Current Consumption	–	–	12	16	µA
t_{START}	Startup Time	From disabled state to enabled state	–	–	300	µs

Table 56-8. Threshold Selection

Symbol	Parameter	Digital Code	Min	Typ	Max	Unit
V_{T-}	Supply Monitor Threshold	0	–	1.6	–	V
		1	–	1.72	–	
		10	–	1.84	–	
		11	–	1.96	–	
		100	–	2.08	–	
		101	–	2.2	–	
		110	–	2.32	–	
		111	–	2.44	–	
		1000	–	2.56	–	
		1001	–	2.68	–	
		1010	–	2.8	–	
		1011	–	2.92	–	
		1100	–	3.04	–	
		1101	–	3.16	–	
		1110	–	3.28	–	
		1111	–	3.4	–	

Figure 56-3. VDDIO Supply Monitor

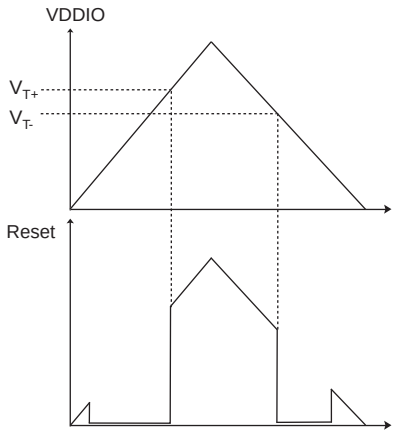


Downloaded from Arrow.com.

Table 56-9. VDDIO Power-On Reset Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{T+}	Threshold Voltage Rising	–	1.47	1.54	1.61	V
V_{T-}	Threshold Voltage Falling	–	1.39	1.46	–	V
V_{hys}	Hysteresis	–	40	80	130	mV
t_{RES}	Reset Time-out Period	–	120	320	800	µs

Figure 56-4. VDDIO Power-On Reset Characteristics



Downloaded from Arrow.com.

56.3 Power Consumption

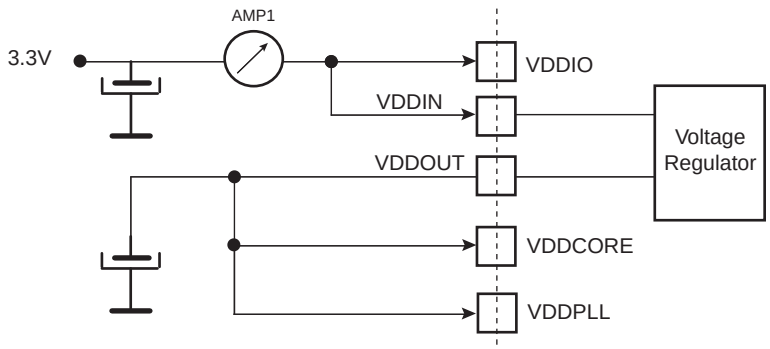
- Power consumption of the device depending on the different Low-Power modes (Backup, Wait, Sleep) and Active mode
- Power consumption on power supply in different modes: Backup, Wait, Sleep and Active
- Power consumption by peripheral:
  - Calculated as the difference in current measurement after enabling then disabling the corresponding clock.
  - Measured when the peripheral is active an doing transfers
- Static and dynamic power consumption of the I/Os

56.3.1 Backup Mode Current Consumption and Wake-Up Time

The Backup mode configurations and measurements are defined as follows:

- Embedded slow clock RC oscillator is enabled
- Supply Monitor on VDDIO is disabled
- RTC is running
- RTT is enabled on 1 Hz mode
- BOD is disabled
- One WKUPx enabled
- Current measurement on AMP1 (see Figure 56-5) with and without the 1 Kbyte backup SRAM
- Measurements are made at ambient temperature

Figure 56-5. Measurement Setup



Downloaded from Arrow.com.

Table 56-10. Typical Power Consumption for Backup Mode with 1 Kbyte Backup SRAM ON

Total Consumption	Typical Value			Unit
	at 25°C	at 85°C	at 105°C	
Conditions	AMP1	AMP1	AMP1	
V_{DDIO} = 3.6V	5.8	18.2	30	µA
V_{DDIO} = 3.3V	4.7	16	27.5	
V_{DDIO} = 3.0V	4.3	15.2	26.3	
V_{DDIO} = 2.5V	3.8	14.1	24.8	
V_{DDIO} = 1.8V	2.8	12.4	22.7	

Table 56-11. Typical Power Consumption for Backup Mode with 1 Kbyte Backup SRAM OFF

Total Consumption	Typical Value			Unit
	at 25°C	at 85°C	at 105°C	
Conditions	AMP1	AMP1	AMP1	
V_{DDIO} = 3.6V	4	11.2	18.2	µA
V_{DDIO} = 3.3V	2.8	8.9	15.1	
V_{DDIO} = 3.0V	2.5	8	13.8	
V_{DDIO} = 2.5V	2.1	7	12.4	
V_{DDIO} = 1.8V	1.1	5.4	10.3	

Table 56-12. Worst Case Power Consumption for Backup Mode with 1 Kbyte Backup SRAM ON

Total Consumption	Worst Case Value			Unit
	at 25°C	at 85°C	at 105°C	
Conditions	AMP1	AMP1	AMP1	
V_{DDIO} = 3.6V	7.1	23.9	43	µA
V_{DDIO} = 3.3V	6	21.7	38	
V_{DDIO} = 3.0V	5.6	20.9	36.8	
V_{DDIO} = 2.5V	6.5	19.5	35.3	
V_{DDIO} = 1.8V	5.5	17.2	31.7	

Table 56-13. Worst Case Power Consumption for Backup Mode with 1 Kbyte Backup SRAM OFF

Total Consumption	Worst Case Value			Unit
	at 25°C	at 85°C	at 105°C	
Conditions	AMP1	AMP1	AMP1	
V_{DDIO} = 3.6V	5	13.8	22	µA
V_{DDIO} = 3.3V	3.5	11	18.1	
V_{DDIO} = 3.0V	3.1	9.9	16.6	
V_{DDIO} = 2.5V	2.6	8.8	15	
V_{DDIO} = 1.8V	1.6	7	12.6	

Downloaded from Arrow.com.

56.3.2 Sleep Mode Current Consumption and Wake-up Time

The Sleep mode configuration and measurements are defined as follows:

- Core clock OFF
- $V_{DDIO} = V_{DDIN} = 3.3V$
- Master Clock (MCK) running at various frequencies with PLLA or the fast RC oscillator
- Fast start-up through WKUP0–13 pins
- Current measurement as shown in Figure 56-6 and associated wake-up time (1)
- All peripheral clocks deactivated
- $T_A = 25°C$

Note: 1. Wake-up time is defined as the delay between the WKUP event and the execution of the first instruction.

Figure 56-6. Measurement Setup for Sleep Mode

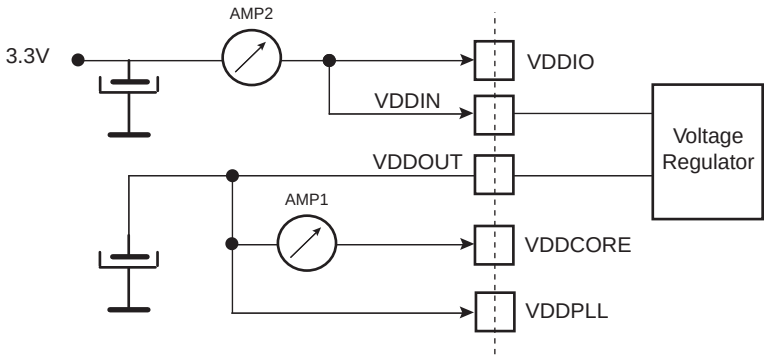


Table 56-14 and Table 56-15 give current consumption and wake-up time in Sleep mode.

Downloaded from Arrow.com.

Table 56-14. Typical Sleep Mode Current Consumption versus Master Clock (MCK) Variation with PLLA

Core Clock/MCK (MHz)	VDDCORE Consumption	Total Consumption	Unit	Wake-up Time	Unit
	AMP1	AMP2			
300/150	20	24	mA	0.85	µs
250/125	17	20		1.05	
150/150	20	24		0.9	
96/96	12.5	15		1.4	
96/48	7.5	10		2.5	
48/48	7	9.5		2.8	
24/24	3.5	5		5.6	
24/12	2	3		10	
12/12	2	3		11.2	
8/8	1.5	2		16.8	
4/4	1.0	1.5		32.9	
4/2	0.9	1		60	
4/1	0.8	1		112.6	

Table 56-15. Typical Sleep Mode Current Consumption versus Master Clock (MCK) Variation with Fast RC Oscillator

Core Clock/MCK (MHz)	VDDCORE Consumption	Total Consumption	Unit	Wake-up Time	Unit
	AMP1	AMP2			
12	2.0	2.0	mA	12	µs
8	1.5	1.5		18	
4	1.0	1.1		31	
2	0.8	0.8		62	
1	0.6	0.7		123	
0.5	0.6	0.6		247	
0.25	0.5	0.5		494	

Downloaded from Arrow.com.

56.3.3 Wait Mode Current Consumption and Wake-up Time

The Wait mode configuration and measurements are defined as follows:

- Core clock and Master clock stopped
- Current measurement as shown in Figure 56-7
- All peripheral clocks deactivated
- BOD disabled
- RTT enabled

Figure 56-7. Measurement Setup for Wait Mode

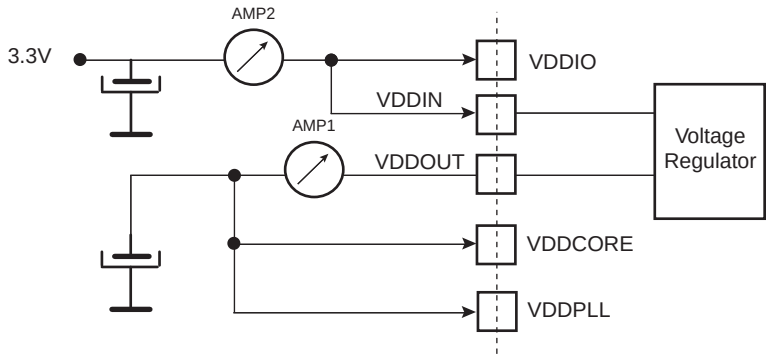


Table 56-16 and Table 56-17 give current consumption and wake-up time[1] in Wait mode.

Table 56-16. Typical Current Consumption in Wait Mode

Wait Mode Consumption	Typical Value				Unit
	at 25°C		at 85°C	at 105°C	
	VDDIO=3.3V		VDDIO=3.3V	VDDIO=3.3V	
Conditions	VDDOUT Consumption AMP1	Total Consumption AMP2	Total Consumption AMP2	Total Consumption AMP2	
No activity on the I/Os of the device	–	0.5	–	7.5	mA

Table 56-17. Typical Wake-up Time to Resume from Wait Mode [1]

Conditions	Wake-up Time from Wait Mode	Unit
Resume from internal Flash with Cache enabled	8.1	µs
Resume from internal Flash with Cache disabled	8.5	µs
Resume from internal SRAM with Cache disabled	8	µs

Note: 1. Wake-up time is defined as the delay between the WKUP event and the execution of the first instruction.

Downloaded from Arrow.com.

56.3.4 Active Mode Power Consumption

The conditions for measurement are defined as follows:

- $V_{DDIO} = V_{DDIN} = 3.3V$
- V_{DDCORE} is provided by the Internal Voltage Regulator
- $T_A = 25°C$
- Application running from Flash memory with 128-bit access mode
- All peripheral clocks are deactivated.
- Master Clock (MCK) running at various frequencies with PLLA or the fast RC oscillator.
- Current measurement on AMP1 (VDDCORE) and total current on AMP2

Figure 56-8. Active Mode Measurement Setup

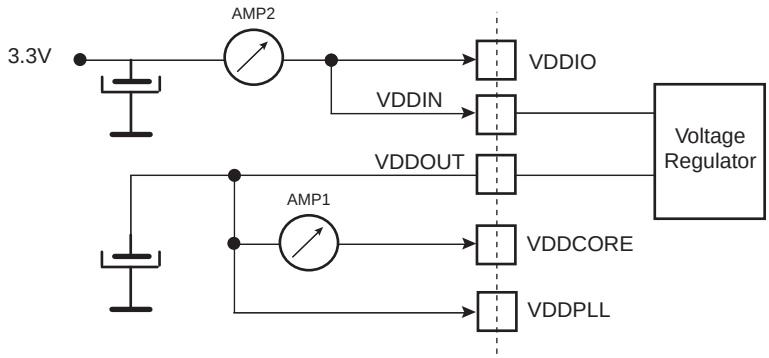


Table 56-18 gives current consumption in Active mode in typical conditions.

Downloaded from Arrow.com.

Table 56-18. Typical Total Active Power Consumption with VDDCORE at 1.2V Running from Embedded Memory (VDDIO + VDDIN - AMP2)

Core Clock/MCK (MHz)	Cortex-M7 Running CoreMark			Unit
	Flash			
	Cache Enable (CE) CoreMark = 4.9/MHz	Cache Disable (CD) CoreMark = 1.0/MHz	TCM CoreMark = 5.0/MHz	
300/150	90	57	83	mA
250/125	77	48	70	
150/150	52	40	48	
96/96	35	27	33	
96/48	31	20	28	
48/48	18	15	17	
24/24	10	8	9	
24/12	9	6	8	
12/12	5	4	5	
8/8	4	3	4	
4/4	2	2	2.5	
4/2	2	1.5	2	
4/1	1.5	1.5	1.5	
2/2	1.5	1.5	1.5	

Note: 1. Flash Wait State (FWS) in EEFC_FMR is adjusted depending on core frequency.

Downloaded from Arrow.com.

56.3.5 Peripheral Power Consumption in Active Mode

The conditions for measurement are defined as follows:

- $V_{DDIO} = V_{DDIN} = 3.3V$
- V_{DDCORE} is provided by the Internal Voltage Regulator
- $T_A = 25°C$
- Peripheral clock is enabled

Table 56-19. Typical Peripheral Power Consumption in Active Mode

Peripheral	Consumption on VDDCORE	Unit
PIO Controller A (PIOA)	4.8	µA/MHz
PIO Controller B (PIOB)	1.5	µA/MHz
PIO Controller C (PIOC)	2.3	µA/MHz
PIO Controller D (PIOD)	3.6	µA/MHz
PIO Controller E (PIOE)	0.4	µA/MHz
UART	1.1	µA/MHz
USART	3.7	µA/MHz
PWM	7.3	µA/MHz
TWIHS	2.4	µA/MHz
SPI	0.7	µA/MHz
Timer Counter (TCx)	1.8	µA/MHz
AFEC	1.6	µA/MHz
DACC	4.9	µA/MHz
ACC	0.1	µA/MHz
HSMCI	6.7	µA/MHz
SMC	4.4	µA/MHz
USBHS	6.8	µA/MHz
GMAC	31.0	µA/MHz
AES	6.3	µA/MHz
XDMAC	14.0	µA/MHz
SSC	2.0	µA/MHz
TRNG	0.3	µA/MHz
ICM	4.6	µA/MHz
ISI	3.1	µA/MHz
QSPI	2.3	µA/MHz
SDRAMC	3.4	µA/MHz
MCAN	9.5	µA/MHz

Downloaded from Arrow.com.

56.3.6 I/O Switching Power Consumption

I/O switching power consumption on VDDCORE and VDDIO are given by the following formulae:

$Power_{VDDCORE} = (DC_{CORE} + AC_{CORE} \times f_{SW}) \times V_{DDCORE}$

$Power_{VDDIO} = (DC_{IO} + AC_{IO_INTERNAL} \times f_{SW} + AC_{IO_EXTERNAL} \times C_{LOAD} \times f_{SW}) \times V_{DDIO}$

where:

- DC_{CORE} is the static power consumption of cells powered by VDDCORE
- AC_{CORE} is the dynamic power consumption of cells powered by VDDCORE
- f_{SW} is the I/O switching frequency
- DC_{IO} is the static power consumption of cells powered by VDDIO
- $AC_{IO_INTERNAL}$ is the dynamic power consumption of internal cells powered by VDDIO
- $AC_{IO_EXTERNAL}$ is the dynamic power consumption due to external load powered by VDDIO
- C_{LOAD} is the overall capacitance on the I/O pin

  $C_{LOAD} = C_{PACKAGE} + C_{PCB} + C_{EXT}$

  where $C_{PACKAGE}$ is defined in the IBIS model of the device and C_{EXT} is the external load capacitance.

The conditions to apply are:

- $T_A = 25°C$
- $C_{PACKAGE} + C_{PCB} = 2$ pF

Table 56-20. I/O Switching Power Consumption (µW)

I/O Group	Conditions	DC_{CORE} µA	AC_{CORE} µA/MHz	DC_{IO} µA	$AC_{IO_INTERNAL}$ µA/MHz	$AC_{IO_EXTERNAL}$ µA/(pF x MHz)
GPIO, CLOCK	STH, Low drive	0.05	1.40	0.06	13.1	3.33
	STH, High drive	0.05	1.40	0.09	24.1	3.33
	Worst case, Low drive	0.05	1.65	0.07	14.6	3.64
	Worst case, High drive	0.05	1.65	0.09	30.6	3.64
GPIO_CLK	STH, Low drive	0.02	0.65	0.06	36.6	3.32
	STH, High drive	0.02	0.65	0.06	62.5	3.33
	Worst case, Low drive	0.02	0.76	0.04	41.6	3.63
	Worst case, High drive	0.02	0.76	0.04	78.8	3.62
GPIO_AD	STH, Low drive	0.02	0.65	0.05	23.5	3.32
	STH, High drive	0.02	0.65	0.05	36.1	3.33
	Worst case, Low drive	0.02	0.76	0.05	26.6	3.63
	Worst case, High drive	0.02	0.76	0.05	42.1	3.62
GPIO_MLB	STH	0.02	0.60	0.04	30.4	3.3
	Worst case	0.02	0.70	0.03	39.3	3.55

Downloaded from Arrow.com.

56.4 Oscillator Characteristics

56.4.1 32 kHz RC Oscillator Characteristics

Table 56-21. 32 kHz RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OSC}	RC Oscillator Frequency	–	20	32	57	kHz
t_{START}	Startup Time	–	–	–	100	µs
I_{DDON}	Current Consumption	After startup time	–	540	–	nA

Downloaded from Arrow.com.

56.4.2 4/8/12 MHz RC Oscillator

The 4/8/12 MHz fast RC oscillator is calibrated in production. This calibration can be read through the Get CALIB Bit command (refer to Section 21. “Enhanced Embedded Flash Controller (EEFC)”) and the frequency can be trimmed by software through the PMC.

Table 56-22. 4/8/12 MHz RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{RANGE}	RC Oscillator Frequency Range	(1)	4	–	12	MHz
ACC_4	4 MHz Total Accuracy (2)	4 MHz output selected (1)	-30	–	46	%
		4 MHz output selected (1)(3)	-12	–	6	%
ACC_8	8 MHz Total Accuracy	8 MHz output selected (1)(3)	-12	–	6	%
ACC_{12}	12 MHz Total Accuracy	12 MHz output selected (1)(3)	-12	–	6	%
Duty	Duty Cycle	–	45	50	55	%
t_{START}	Startup Time	–	–	–	20	µs
I_{DDON}	Active Current Consumption(3)	4 MHz	–	50	68	µA
		8 MHz	–	65	86	
		12 MHz	–	82	102	

Notes: 1. Frequency range can be configured in the Supply Controller registers.

2. Not trimmed from factory.

3. After trimming at 25°C and VDDCORE = 1.2V.

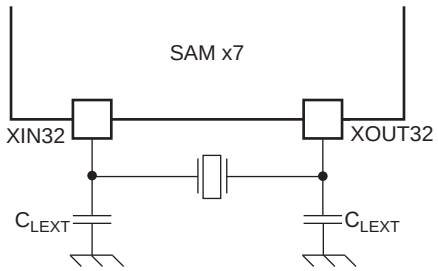
Downloaded from Arrow.com.

56.4.3 32.768 kHz Crystal Oscillator Characteristics

Table 56-23. 32.768 kHz Crystal Oscillator Characteristics

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
f_{OSC}	Operating Frequency	Normal mode with crystal		–	32.77	–	kHz
t_{START}	Startup Time	50 800 slow clock cycles		0.9	–	2.4	s
I_{DDON}	Current Consumption	ESR < 50 kOhm	$C_{CRYSTAL}$ = 12.5 pF	–	440	680	nA
			$C_{CRYSTAL}$ = 6 pF		300	650	
		ESR < 100 kOhm	$C_{CRYSTAL}$ = 12.5 pF		450	650	
			$C_{CRYSTAL}$ = 6 pF		450	750	
C_{PARA}	Internal Parasitic Capacitance	–		0.4	0.5	0.6	pF

Figure 56-9. 32.768 kHz Crystal Oscillator Schematics



$C_{LEXT} = 2 \times (C_{CRYSTAL} - C_{PARA} - C_{PCB})$

where C_{PCB} is the capacitance of the printed circuit board (PCB) track layout from the crystal to the pin.

Downloaded from Arrow.com.

56.4.4 32.768 kHz Crystal Characteristics

Table 56-24. 32.768 kHz Crystal Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
ESR	Equivalent Series Resistor	Crystal at 32.768 kHz	–	50	100	kOhm
C_M	Motional Capacitance	Crystal at 32.768 kHz	2	–	4	fF
C_{SHUNT}	Shunt Capacitance	Crystal at 32.768 kHz	0.6	–	2	pF
$C_{CRYSTAL}$	Allowed Crystal Capacitance Load	From crystal specification	6	–	12.5	pF
P_{ON}	Drive Level	–	–	–	0.2	µW

56.4.5 XIN32 Clock Characteristics in Bypass Mode

Table 56-25. XIN32 Clock Characteristics In Bypass Mode

Symbol	Parameter	Conditions	Min	Typ	Max
$1/(t_{CPXIN})$	XIN32 Clock Frequency	(1)	–	–	32
t_{CHXIN}	XIN32 Clock High Half-period	(1)	15	–	–
t_{CLXIN}	XIN32 Clock Low Half-period	(1)	15	–	–
V_{XIN_IL}	V_{XIN} Input Low-level Voltage	(1)	Min of V_{IL} for CLOCK pad	–	Max of V_{IL} for CLOCK pad
V_{XIN_IH}	V_{XIN} Input High-level Voltage	(1)	Min of V_{IH} for CLOCK pad	–	Max of V_{IH} for CLOCK pad

Note: 1. These characteristics apply only when the 32.768 kHz crystal oscillator is in Bypass mode.

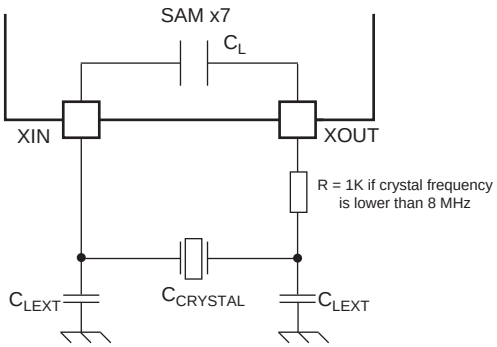
Downloaded from Arrow.com.

56.4.6 3 to 20 MHz Crystal Oscillator Characteristics

Table 56-26. 3 to 20 MHz Crystal Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OSC}	Operating Frequency	Normal mode with crystal	3	–	20	MHz
t_{START}	Startup Time	3 MHz, C_{SHUNT} = 3 pF	–	–	40	ms
		12 MHz, C_{SHUNT} = 7 pF with C_M = 1.6 fF	–	–	6	ms
		20 MHz, C_{SHUNT} = 7 pF with C_M = 1.6 fF	–	–	5.7	ms
I_{DDON}	Current Consumption (on VDDIO)	3 MHz	–	230	470	µA
		12 MHz	–	390	600	µA
		20 MHz	–	450	750	µA
C_L	Internal Equivalent Load Capacitance	Integrated Load Capacitance (XIN and XOUT in series)	7.5	9	10.5	pF

Figure 56-10. 3 to 20 MHz Crystal Oscillator Schematics



$C_{LEXT} = 2 \times (C_{CRYSTAL} - C_L - C_{PCB})$

where C_{PCB} is the capacitance of the printed circuit board (PCB) track layout from the crystal to the pin.

Downloaded from Arrow.com.

56.4.7 3 to 20 MHz Crystal Characteristics

Table 56-27. 3 to 20 MHz Crystal Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
ESR	Equivalent Series Resistor	Fundamental at 3 MHz	–	–	150	Ohm
		Fundamental at 8 MHz			140	
		Fundamental at 12 MHz			120	
		Fundamental at 16 MHz			80	
		Fundamental at 20 MHz			50	
C_M	Motional Capacitance	Fundamental at 3 MHz	3	–	8	fF
		Fundamental at 8–20 MHz	1.6	–	8	
C_{SHUNT}	Shunt Capacitance	–	–	–	7	pF
$C_{CRYSTAL}$	Allowed Crystal Capacitance Load	From crystal specification	12.5	–	17.5	pF
P_{ON}	Drive Level	3 MHz	–	–	15	µW
		8 MHz	–	–	30	
		12 MHz, 20 MHz	–	–	50	

Downloaded from Arrow.com.

56.4.8 3 to 20 MHz XIN Clock Input Characteristics in Bypass Mode

Table 56-28. 3 to 20 MHz XIN Clock Input Characteristics in Bypass Mode

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CPXIN})$	XIN Clock Frequency	(1)	–	–	20	MHz
t_{CHXIN}	XIN Clock High Half-period	(1)	25	–	–	ns
t_{CLXIN}	XIN Clock Low Half-period	(1)	25	–	–	ns
V_{XIN_IL}	V_{XIN} Input Low-level Voltage	(1)	Min of V_{IL} for CLOCK pad	–	Max of V_{IL} for CLOCK pad	V
V_{XIN_IH}	V_{XIN} Input High-level Voltage	(1)	Min of V_{IH} for CLOCK pad	–	Max of V_{IH} for CLOCK pad	V

Note: 1. These characteristics apply only when the 3–20 MHz crystal oscillator is in Bypass mode.

Downloaded from Arrow.com.

56.4.9 Crystal Oscillator Design Considerations

56.4.9.1 Choosing a Crystal

When choosing a crystal for the 32768 Hz Slow Clock Oscillator or for the 3–20 MHz oscillator, several parameters must be taken into account. Important parameters between crystal and product specifications are as follows:

- Crystal Load Capacitance
  - The total capacitance loading the crystal, including the oscillator's internal parasitics and the PCB parasitics, must match the load capacitance for which the crystal's frequency is specified. Any mismatch in the load capacitance with respect to the crystal's specification will lead to inaccurate oscillation frequency
- Drive Level
  - Crystal drive level ≥ Oscillator Drive Level. Having a crystal drive level number lower than the oscillator specification may damage the crystal.
- Equivalent Series Resistor (ESR)
  - Crystal ESR ≤ Oscillator ESR Max. Having a crystal with ESR value higher than the oscillator may cause the oscillator to not start.
- Shunt Capacitance
  - Max. crystal shunt capacitance ≤ Oscillator Shunt Capacitance (C_{SHUNT}). Having a crystal with C_{SHUNT} value higher than the oscillator may cause the oscillator to not start.

56.4.9.2 Printed Circuit Board (PCB)

To minimize inductive and capacitive parasitics associated with XIN, XOUT, XIN32 and XOUT32 nets, it is recommended to route them as short as possible. Additionally, it is of prime importance to keep these nets away from noisy switching signals (clock, data, PWM, etc.). A good practice is to shield them with a quiet ground net to avoid coupling to neighboring signals.

Downloaded from Arrow.com.

56.5 PLLA Characteristics

Table 56-29. PLLA Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{IN}	Input Frequency	–	8	–	32	MHz
f_{OUT}	Output Frequency	–	160	–	500	MHz
I_{PLL}	Current Consumption	Active mode at 160 MHz at 1.2V	–	2.2	3	mA
		Active mode at 500 MHz at 1.2V	–	8	10	
t_{START}	Startup Time	–	–	–	300	µs

56.6 PLLUSB Characteristics

Table 56-30. PLLUSB Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{IN}	Input Frequency	–	–	(1)	–	MHz
f_{OUT}	Output Frequency	–		480		MHz
I_{PLLUSB}	Current Consumption	In Active mode, on VDDPLLUSB	–	4.9	6.4	mA
		In Active mode, on VDDCORE	–	–	1	mA
t_{START}	Startup Time	–	–	–	50	µs

Note: 1. 12 or 16 MHz. Refer to UTMI_CKTRIM register for clock selection.

Downloaded from Arrow.com.

56.7 USB Transceiver Characteristics

The device conforms to all voltage, power, and timing characteristics and specifications as set forth in the USB 2.0 Specification. Refer to the USB 2.0 Specification for more information.

Table 56-31. USB Transceiver Dynamic Power Consumption

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
I_{BIAS}	Bias Current Consumption on VBG	–	–	–	12	mA
$I_{VDDUTMII}$	HS Transceiver Current Consumption	HS transmission	–	–	44	mA
	HS Transceiver Current Consumption	HS reception	–	–	24	mA
	LS / FS Transceiver Current Consumption	FS transmission 0m cable(1)	–	–	5	mA
	LS / FS Transceiver Current Consumption	FS transmission 5m cable(1)	–	–	30	mA
	LS / FS Transceiver Current Consumption	FS reception(1)	–	–	1	mA
$I_{VDDUTMIC}$	Core	–	–	–	10	mA

Note: 1. Including 1 mA due to pull-up/pull-down current consumption.

Downloaded from Arrow.com.

56.8 AFE Characteristics

Electrical data are in accordance with an operating temperature range from -40°C to +105°C unless otherwise specified.

VREFP is the positive reference of the AFE. The VREFN pin must be connected to ground.

DAC1 and DAC0 provide an analog output voltage (V_{DAC}) in the range [0 : VREFP] with an accuracy equal to 10 bits. The DAC output voltage is single-ended and is used as a reference node by the sampling stage S/H0 and S/H1 (Sample-and-Hold PGA), relative to the single-ended input signal being sampled on the selected channel.

As a consequence, programming the DAC output voltage offers a capability to compensate for a DC offset on the input signal being sampled. DC offset compensation is effective in single-ended operation and is not effective in fully differential operation.

During fully differential operation, the DAC10 output voltage can be programmed at VREFP/2, by using the 10-bit code 512. The DAC value does not affect the AFE output code.

VREFP/2 on DAC0 and DAC1 is not automatically set and must be programed as the code 512 into the channel corresponding DAC0 and DAC1.

Figure 56-11 and Figure 56-12 illustrate the architecture of the AFE in Single-ended and in Differential modes.

Downloaded from Arrow.com.

Figure 56-11. Single-ended Mode AFE

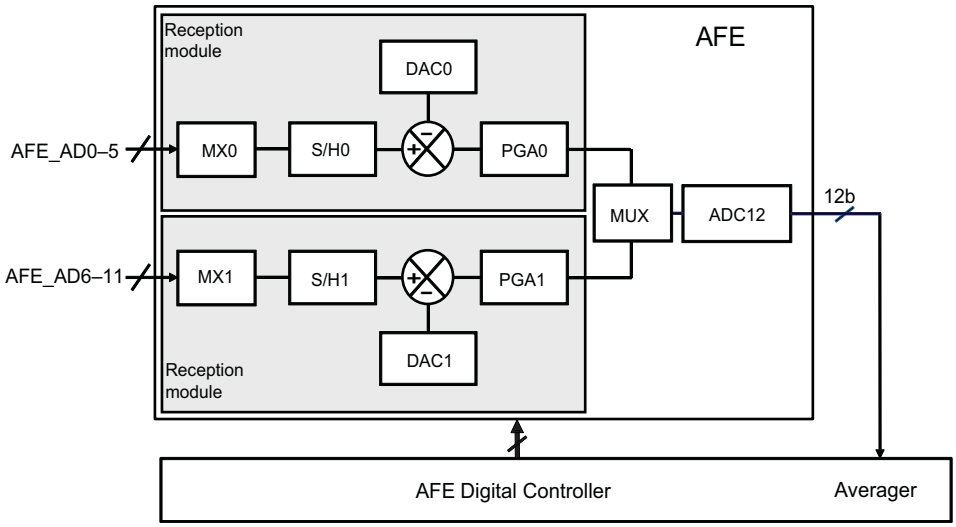
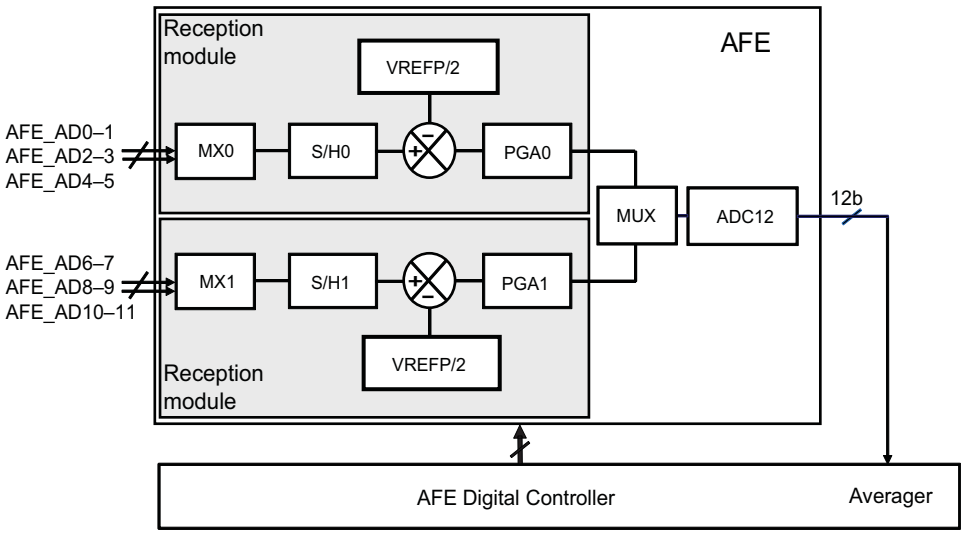


Figure 56-12. Differential Mode AFE



Downloaded from Arrow.com.

56.8.1 AFE Power Supply

56.8.1.1 Power Supply Characteristics

Table 56-32. Power Supply Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
I_{VDDIN}	Analog Current Consumption	Sleep mode [2] Fast wake-up mode [3] Normal mode, single sampling Normal mode, dual sampling	–	2 0.4 3.4 4.2	4 0.6 4.6 5.8	µA mA mA mA
$I_{VDDCORE}$	Digital Current Consumption	Sleep mode [2] Normal mode	-	1 80	2 100	µA µA

Notes:
1. Current consumption is measured with AFEC_ACR.IBCTL=1.
2. In Sleep mode, the AFE core, the Sample and Hold and the internal reference operational amplifer are off.
3. In Fast Wake-up mode, only the AFE core is off.

56.8.1.2 ADC Bias Current

AFE_ACR.IBCTL controls the ADC bias current, with the nominal setting IBCTL = 1.

IBCTL = 1 is the default configuration suitable for a sampling frequency of up to 1 MHz. If the sampling frequency is below 500 kHz, IBCTL = 0 can also be used to reduce the current consumption.

Downloaded from Arrow.com.

56.8.2 External Reference Voltage

V_{VREFP} is an external reference voltage applied on the pin VREFP. The quality of the reference voltage V_{VREFP} is critical to the performance of the AFE. A DC variation of the reference voltage V_{VREFP} is converted to a gain error by the AFE. The noise generated by V_{VREFP} is converted by the AFE to count noise.

Table 56-33. VREFP Electrical Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{VREFP}	Voltage Range	Full operational	1.7	–	VDDIN	V
	RMS Noise	Bandwidth 10 kHz to 1 MHz	–	–	100	µV
R_{VREFP}	Input DC Impedance	AFE reference resistance bridge(1)	3.9	4.7	5.4	kOhm
I_{VREFP}	Current	V_{VREFP}=3.3V	-	0.8	1	mA

Note: 1. When the AFE is in Sleep mode, the VREFP impedance has a minimum of 10 MOhm.

Downloaded from Arrow.com.

56.8.3 AFE Timings

Table 56-34. AFE Timing Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$f_{AFE\ Clock}$	Clock Frequency	–	4	20	40	MHz
$t_{AFE\ Clock}$	Clock Period	–	25	50	250	ns
f_S	Sampling Frequency(2)	–	–	–	2	MHz
t_{START}	AFE Startup Time	Sleep mode to Normal mode Fast Wake-up mode to Normal mode	–	–	4 2	µs
$t_{TRACKTIM}$	Tracking Time	If AFEC_MR.TRACKTIM is programmed < 15, then the min value is applied by default	15	–	–	$t_{AFE\ Clock}$
t_{CONV}	Conversion Time	Number of 12-bit ADC clock pulses to perform a conversion	–	15	–	$t_{AFE\ Clock}$
$t_{TRANSFER}$ (1)	AFEC_MR.TRANSFER=2 AFEC_MR.TRANSFER=1 AFEC_MR.TRANSFER=0	If 35 MHz< $f_{AFE\ Clock}$ <40 MHz If 30 MHz< $f_{AFE\ Clock}$ <35 MHz If 4 MHz< $f_{AFE\ Clock}$ <30 MHz	8 7 6	–	–	$t_{AFE\ Clock}$

Notes:
1. Number of AFE clock pulses to transfer the S&H voltage to the ADC core.
2. $f_S=1/t_{AFE_conv}$ in Free Run mode; otherwise defined by the trigger timing.

Downloaded from Arrow.com.

56.8.4 AFE Transfer Function

The first operation of the AFE is a sampling function relative to V_{DAC}. V_{DAC} is generated by an internal DAC0 or DAC1. All operations after the Sample-and-Hold are differential relative to an internal common mode voltage $V_{CM} = V_{VREFP}/2$.

In Differential mode, the Sample-and-Hold common mode voltage is equal to $V_{DAC} = V_{VREFP}/2$ (set by software DAC0 and DAC1 to code 512).

In Single-ended mode, V_{DAC} is the common mode voltage. V_{DAC} is the output of DAC0 or DAC1 voltage. All operations after the Sample-and-Hold are differential, including those in Single-ended mode.

For the formula example, the internal DAC0 or DAC1 is set for the code 512.

The DATA code in AFEC_CDR is up to 16-bit positive integer or two's complement (signed integer).

The code does not exceed 4095 when the field AFEC_EMR.RES=0 (12-bit mode, no averaging).

56.8.4.1 Differential Mode (12-bit mode)

A differential input voltage $V_{IN} = V_{INP} - V_{INN}$ can be applied between two selected differential pins, e.g. AFE0_AD0 and AFE0_AD1.The ideal code C_i is calculated by using the following formula and rounding the result to the nearest positive integer.

$$C_i = \frac{4096}{V_{VREFP}} \times V_{IN} \times Gain + (2047)$$

For the other resolution defined by RES, the code C_i is extended to the corresponding resolution.

Table 56-35 is a computation example for the above formula, where V_{VREFP} = 3V.

Table 56-35. Input Voltage Values in Differential Mode, Non-signed Output

C_i		Gain		
Signed	**Non-signed**	**1**	**2**	**4**
-2048	0	-3	-1.5	-0.75
0	2047	0	0	0
2047	4095	3	1.5	0.75

56.8.4.2 Single-ended Mode (12-bit mode)

A single input voltage V_{IN} can be applied to selected pins, e.g. AFE0_AD0 or AFE0_AD1. The ideal code C_i is calculated using the following formula and rounding the result to the nearest positive integer.

The single-ended ideal code conversion formula is:

$$C_i = \frac{4096}{V_{VREFP}} \times (V_{IN} - V_{DAC}) \times Gain + 2047$$

For the other resolution defined by RES, the code C_i is extended to the corresponding resolution.

Table 56-36 is a computation example for the above formula, where V_{VREFP} = 3V:

Table 56-36. Input Voltage Values in Single-ended Mode

C_i		Gain		
Signed	**Non-signed**	**1**	**2**	**4**
-2048	0	0	0.75	1.125
0	2047	1.5	1.5	1.5
2047	4095	3	2.25	1.875

Downloaded from Arrow.com.

56.8.4.3 Example of LSB Computation

The LSB is relative to the analog scale V_{VREFP}.

The term LSB expresses the quantization step in volts, also used for one AFE code variation.

- Single-ended (SE) (ex: V_{VREFP} = 3.0V)
  - Gain = 1, LSB = (3.0V / 4096) = 732 µV
  - Gain = 2, LSB = (1.5V / 4096) = 366 µV
  - Gain = 4, LSB = (750 mV / 4096) = 183 µV
- Differential (DIFF) (ex: V_{VREFP} = 3.0V)
  - Gain = 0.5, LSB = (6.0V / 4096) = 1465 µV
  - Gain = 1, LSB = (3.0V / 4096) = 732 µV
  - Gain = 2, LSB = (1.5V / 4096) = 366 µV

The data include the AFE performances, as the PGA and AFE core cannot be separated. The temperature and voltage dependency are given as separate parameters.

Downloaded from Arrow.com.

56.8.4.4 Gain and Offset Errors

For:

- a given gain error: E_G (%)
- a given ideal code (C_i)
- a given offset error: E_O (LSB of 12 bits)

in 12-bit mode, the actual code (C_A) is calculated using the following formula

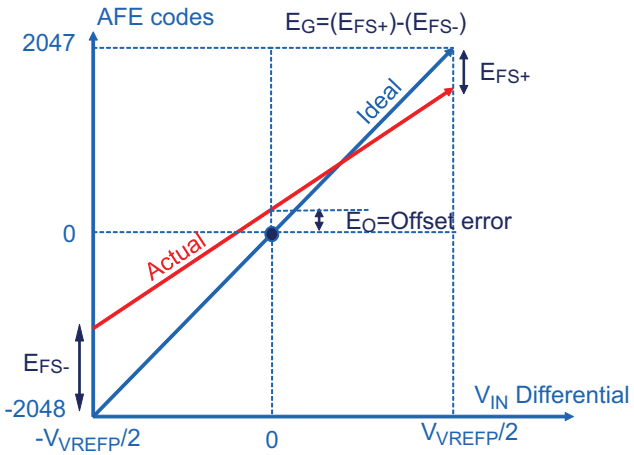
$$C_A = \left(1 + \frac{E_G}{100}\right) \times (C_i - 2047) + 2047 + E_O$$

For higher resolutions, the code can be extended to the corresponding resolution defined by RES.

Differential Mode

In Differential mode, the offset is defined when the differential input voltage is zero.

Figure 56-13. Gain and Offset Errors in Differential Mode



where:

- Full-scale error E_{FS} =(E_{FS+})-(E_{FS-}), unit is LSB code
- Offset error E_O is the offset error measured for V_{IN}=0V
- Gain error E_G=100 × E_{FS} /4096, unit in %

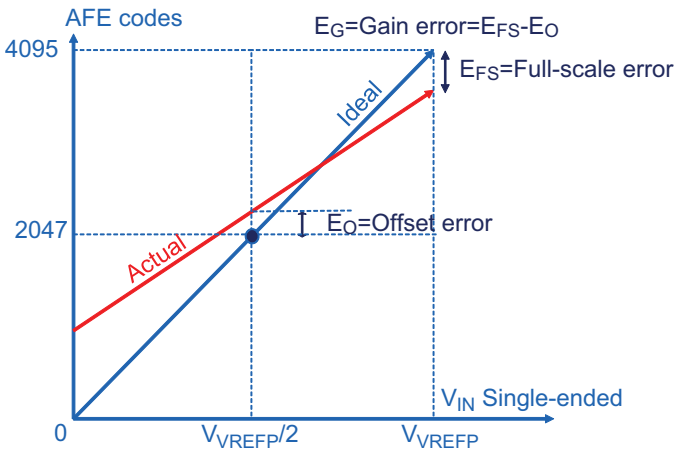
The error values in Table 56-38 include the sample-and-hold error as well as the PGA gain error.

Downloaded from Arrow.com.

Single-ended Mode

Figure 56-14 illustrates the AFE output code relative to an input voltage V_{IN} between 0V (Ground) and V_{VREFP}. The AFE is configured in Single-ended mode by connecting internally the negative differential input to $V_{VREFP}/2$. As the AFE continues to work internally in Differential mode, the offset is measured at $V_{VREFP}/2$. The offset at $V_{INP}=0$ can be computed using the transfer function and the corresponding E_G and E_O.

Figure 56-14. Gain and Offset Errors in Single-ended Mode



where:

- Full-scale error $E_{FS}=(E_{FS+})-(E_{FS-})$, unit is LSB code
- Offset error E_O is the offset error measured for $V_{INP}=0V$
- Gain error $E_G=100 \times E_{FS}/4096$, unit in %

The error values in Table 56-38 include the DAC, the sample-and-hold error as well as the PGA gain error.

Downloaded from Arrow.com.

56.8.5 AFE Electrical Characteristics

Table 56-37. AFE INL and DNL, V_{VREFP} = 3.3V

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
		Differential Mode				
INL	Integral Non-Linearity	For T_A [-40°C : 25°C] For T_A [25°C : 70°C] For T_A [70°C : 105°C]	**-2** -4 -9	–	2 4 9	LSB
DNL	Differential Non-Linearity	For T_A [-40°C : 25°C] For T_A [25°C : 70°C] For T_A [70°C : 105°C]	-1.5 -2 -2.5	–	1.5 2 2.5	LSB
		Single-Ended Mode				
INL	Integral Non-Linearity	For T_A [-40°C : 25°C] For T_A [25°C : 70°C] For T_A [70°C : 105°C]	**-2.5** -7 -12	–	2.5 7 12	LSB
DNL	Differential Non-Linearity	For T_A [-40°C : 25°C] For T_A [25°C : 70°C] For T_A [70°C : 105°C]	-1.5 -2 -2.5	–	1.5 2 2.5	LSB

Table 56-38. AFE Offset and Gain Error, V_{VREFP} = 3.3V and GAIN = 1

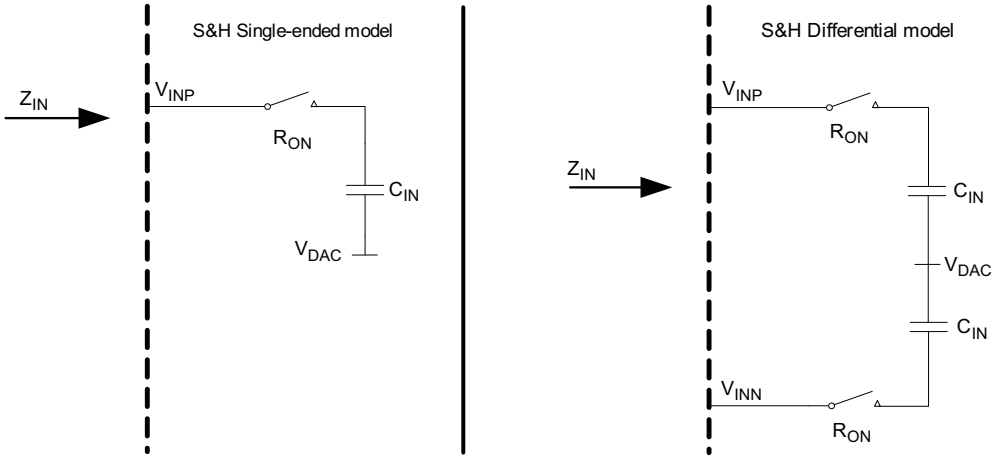
Symbol	Parameter	Conditions	Min	Typ (1)	Max	Unit
		Differential Mode				
E_O	Differential Offset Error (1)	For T_A [-40°C : 25°C] For T_A [25°C : 70°C] For T_A [70°C : 105°C]	-20 -60 -80	–	20 60 80	LSB
E_G	Differential Gain Error	For T_A [-40°C : 25°C] For T_A [25°C : 70°C] For T_A [70°C : 105°C]	-2.7 -3 -4	–	2.7 3 4	%
		Single-Ended Mode				
E_O	Single-ended Offset Error (1)	For T_A [-40°C : 25°C] For T_A [25°C : 70°C] For T_A [70°C : 105°C]	-20 -60 -80	–	20 60 80	LSB
E_G	Single-ended Gain Error	For T_A [-40°C : 25°C] For T_A [25°C : 70°C] For T_A [70°C : 105°C]	-2.7 -3 -4	–	2.7 3 4	%

Note: 1. For higher Gain values, AFE offset = E_O x GAIN.

Downloaded from Arrow.com.

56.8.6 AFE Channel Input Impedance

Figure 56-15. Input Channel Model



where:

- Z_{IN} is input impedance in Single-ended or Differential mode
- C_{IN} = 2 to 8 pF ±20% depending on the gain value and mode (SE or DIFF); temperature dependency is negligible
- R_{ON} is typical 2 kΩ and 8 kΩ max (worst case process and high temperature)

The following formula is used to calculate input impedance:

$$Z_{IN} = \frac{1}{f_S \times C_{IN}}$$

where:

- f_S is the sampling frequency of the AFE channel
- Typ values are used to compute AFE input impedance Z_{IN}

Table 56-39. Input Capacitance (C_{IN}) Values

Gain Selection	Single-ended	Differential	Unit
1	2	2	pF
2	4	4	
4	8	8	

Table 56-40. Z_{IN} Input Impedance

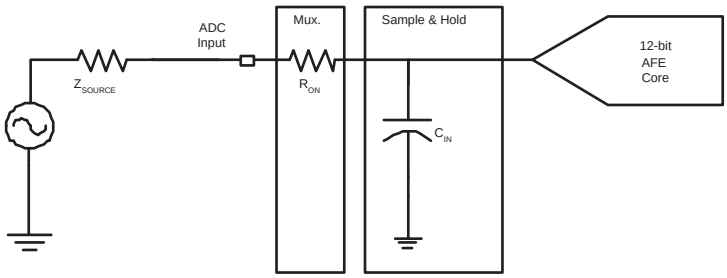
f_S (MHz)	1	0.5	0.25	0.125	0.0625	0.03125	0.015625	0.007813
C_{IN} = 2 pF								
Z_{IN} (MΩ)	0.5	1	2	4	8	16	32	64
C_{IN} = 4 pF								
Z_{IN} (MΩ)	0.25	0.5	1	2	4	8	16	32
C_{IN} = 8 pF								
Z_{IN} (MΩ)	0.125	0.25	0.5	1	2	4	8	16

Downloaded from Arrow.com.

Track and Hold Time versus Source Output Impedance

Figure 56-16 shows a simplified acquisition path.

Figure 56-16. Simplified Acquisition Path



During the tracking phase, the AFE tracks the input signal during the tracking time shown below:

$$t_{TRACK} = n \times C_{IN} \times (R_{ON}+Z_{SOURCE})/1000$$

- Tracking time expressed in ns and Z_{SOURCE} expressed in Ω.
- n depends on the expected accuracy
- R_{ON}=2 kOhm

Table 56-41. Number of Tau:n

Resolution (bits)	12	13	14	15	16
RES	0	2	3	4	5
n	8	9	10	11	12

The AFE already includes a tracking time of 15 $t_{AFE\ Clock.}$ As a result, two cases can be considered:

- If the calculated tracking time is lower than 15 $t_{AFE\ Clock}$, then AFEC_MR.TRACKTIM can be set to 0.
- If the calculated tracking time is higher than 15 $t_{AFE\ Clock}$, then AFEC_MR.TRACKTIM must be set to the value corresponding to the tracking time computation.

Downloaded from Arrow.com.

56.8.6.1 AFE DAC Offset Compensation

Table 56-42. DAC Static Performances[1]

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
N	Resolution [2]	–	8	9	10	LSB
INL	Integral Non Linearity	–	-2	–	2	LSB
DNL	Differential Non Linearity	–	-1	–	1	LSB

Notes:
1. DAC Offset is included in the AFE E_O performances.
2. 10 bits LSB relative to VREFP scale, LSB = $V_{VREFP}/2^{10}$ = 2.93 mV, with V_{VREFP} = 3V.

Downloaded from Arrow.com.

56.8.7 AFE Resolution with Averaging

For measurements in Table 56-43, the following conditions apply:

- In typical conditions of temperature and process
- External Trigger mode
- Gain = 1
- $f_{AFE\ Clock}$ = 10 MHz
- f_S = 267 kHz/OSR
- V_{REFP} = 3.3V
- Signal frequency = 1 kHz

Table 56-43. AFE Resolution following Digital Averaging

RES[1]	OSR	Conditions	Mode[2]	SNR[3]	ENOB	f_S (kHz)
Differential Mode						
0	1	See above.	12	62.0	10.0	267.0
2	4		13	66.2	10.7	66.8
3	16		14	70.3	11.4	16.7
4	64		15	74.5	12.1	4.2
5	256		16	78.7	12.8	1.0
Single-Ended Mode						
0	1	See above.	12	59.9	9.7	267.0
2	4		13	64.3	10.4	66.8
3	16		14	68.6	11.1	16.7
4	64		15	73.0	11.8	4.2
5	256		16	77.4	12.6	1.0

Notes:
1. RES is configured in the register AFEC_EMR.
2. Mode in bits is the word length of the average output defined by the RES field.
3. THD typically -70 dB is not shown because not affected by the averaging.
   When V_{VREFP} = 1.6V, an SNR loss of 6 dB can be expected, leading to an ENOB reduction of 1 bit.
   For Gain=2 and 4, the SNR can be reduced 1 to 2 dB.

Downloaded from Arrow.com.

56.9 Analog Comparator Characteristics

Table 56-44. Analog Comparator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{IR}	Input Voltage Range	–	GND + 0.2	–	V_{DDIN} - 0.2	V
V_{IO}	Input Offset Voltage	–	–	–	10	mV
I_{VDDIN}	Current Consumption (VDDIN)	Low-power option (ACC_ACR.ISEL = 0)	–	20	25	µA
		High-speed option (ACC_ACR.ISEL = 1)	–	120	170	
V_{hys}	Hysteresis	ACC_ACR.HYST = 1 or 2 ACC_ACR.ISEL = 0	–	20	50	mV
		ACC_ACR.HYST = 3 ACC_ACR.ISEL = 0	–	40	90	
		ACC_ACR.HYST = 1 or 2 ACC_ACR.ISEL = 1	–	25	60	mV
		ACC_ACR.HYST = 3 ACC_ACR.ISEL = 1	–	45	110	
t_S	Settling Time	Overdrive > 100 mV (ACC_ACR.ISEL = 0)	–	–	1	µs
		Overdrive > 100 mV (ACC_ACR.ISEL = 1)	–	–	0.1	

Downloaded from Arrow.com.

56.10 Temperature Sensor

The temperature sensor is connected to channel 11 of the AFE0.

The temperature sensor provides an output voltage (V_{TEMP}) that is proportional to absolute temperature (PTAT).

Improvement of the raw performance of the temperature sensor acquisition can be achieved by performing a single temperature point calibration to remove the initial inaccuracies (V_{TEMP} and ADC offsets).

Table 56-45. Temperature Sensor Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{TEMP}	Output Voltage via AD11	T_A = 25°C	0.64	0.72	0.8	V
dV_{TEMP}/dT	Temperature Sensitivity (Slope Voltage versus Temperature)	–	2.11	2.33	2.55	mV/°C
t_S	V_{TEMP} Settling Time	When V_{TEMP} is sampled by the AFEC, the required track-and-hold time to ensure 1°C accurate settling	1	–	–	µs
–	Temperature Accuracy	After offset calibration over T_A range [-40°C : +105°C]	-8	–	8	°C
–	Temperature Accuracy	After offset calibration over T_A range [0°C : +85°C]	-6	–	6	°C
t_{START}	Startup Time	–	–	–	30	µs
I_{VDDIN}	Current Consumption	–	–	130	270	µA

Downloaded from Arrow.com.

56.11 12-bit DAC Characteristics

Table 56-46. Analog Power Supply Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{DDIN}	Analog Supply	–	2	3.3	3.6	V
I_{VDDIN}	Current Consumption	Sleep mode (Clock OFF)	–	10	–	µA
		Normal mode with one output on, DACC_ACR.IBCTLCHx=3(1) F_S = 1 MSps, no R_{LOAD}, V_{DDIN} = 3.3V	–	200	600	
		Normal mode with one output on, DACC_ACR.IBCTLCHx=1(1) F_S = 500 KSps, no R_{LOAD}, V_{DDIN} = 3.3V	–	100	300	
		Bypass mode (output buffer off) with one output on, DACC_ACR.IBCTLCHx=0(1) F_S = 500 KSps, no R_{LOAD}, V_{DDIN} = 3.3V	–	10	20	
PSRR	Power Supply Rejection Ratio (V_{DDIN})	V_{DDIN} ±10 mV Up to 10 kHz	–	70	–	dB

Note: 1. The maximum conversion rate vs the configuration of DACC_ACR.IBCTLHx is given in the table below.

DACC_ACR.IBCTLCHx	Max. Conversion Rate	Type
0	Bypass	D
1	500 ks/s	D
2	N/A	D
3	1 Ms/s	D

Table 56-47. Voltage Reference (1)

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{VREFP}	Positive Voltage Reference	Externally decoupled 1 µF	1.7	1.8	V_{DDIN}	V
I_{VREFP}	DC Current on VREFP	–	–	2.5	5	µA

Note: 1. VREFP is the positive reference shared with AFE and may have a different value for AFE. Refer to the AFE electrical characteristics if AFE is used. The VREFN pin must be connected to ground.

Table 56-48. DAC Clock

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{DAC}	DAC Clock Frequency	–	–	–	12	MHz
f_S	Sampling Frequency	–	–	f_{DAC}/12	–	MHz

Downloaded from Arrow.com.

Table 56-49. Static Performance Characteristics

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
	Resolution	–		–	12	–	bit
INL	Integral Non-linearity	No R_{LOAD} C_{LOAD}=50 pF DACC_ACR.IBCTLCHx=3 V_{VREFP} = 1.7V V_{DDIN} from 2.0V to 2.5V (1)		-7	–	+7	LSB
		No R_{LOAD} C_{LOAD}=50 pF DACC_ACR.IBCTLCHx=3 V_{VREFP} = 1.7V V_{DDIN} from 2.5V to 3.6V (1)		-7	±1	+5	LSB
DNL	Differential Non-linearity	No R_{LOAD} C_{LOAD}=50 pF DACC_ACR.IBCTLCHx=3 V_{VREFP} = 1.7V V_{DDIN} from 2.0V to 2.5V (1)	T_A [-40°C : 105°C]	-3.5	–	+8	LSB
		No R_{LOAD} C_{LOAD}=50 pF DACC_ACR.IBCTLCHx=3 V_{VREFP} = 1.7V V_{DDIN} from 2.5V to 3.6V (1)	T_A = 25°C	-2.5	–	+5	LSB
			T_A [-40°C : 105°C]	-3.5	–	+5	
E_O	Offset Error	V_{DDIN} from 2.0V to 3.6V (2)		-8	–	+8	mV
E_G	Gain Error	No R_{LOAD} C_{LOAD}=50 pF DACC_ACR.IBCTLCHx=3 V_{VREFP} = 1.7V V_{DDIN} from 2.0V to 3.6V		-3	-1	–	%.FSR

Notes:
1. Best-fit Curl from 0x080 to 0xF7F.
2. Difference between DACx at 0x800 and V_{VREFP}/2.

Downloaded from Arrow.com.

Table 56-50. Dynamic Performance Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{START}	Startup Time	From DAC on (CHER.CHx) to DAC ready to convert (CHSR.DACRDYx)	–	–	10	µs
t_S	Settling Time Code to Code; i.e., code(n-1) to code(n) ± 0.5 LSB	R_{LOAD}=5 Kohm C_{LOAD}=50 pF DACC_ACR.IBCTLCHx=3 F_S = 1 MSps	–	0.5	1	µs
	Settling Time Full-scale; i.e., 0x000 to 0xFFF ±0.5 LSB		–	1	2	µs
SINAD	Signal to Noise and Distortion	Differential mode V_{DDIN} = 3.3V F_S = 1 MSps V_{REFP} = 1.7V DACC_ACR.IBCTLCHx=3 F_{IN} = 10 kHz, 0.5V rms_diff Bandwidth = 20 Hz to 24 kHz	60	70	–	dB

Downloaded from Arrow.com.

Table 56-51. Analog Outputs

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
R_{LOAD}	Output Resistor Load	Output load resistor	5	–	–	kOhm
C_{LOAD}	Output Capacitor Load	Output load capacitor	–	–	50	pF
V_{DACx_MIN}	Minimum Output Voltage on DACx	Code = 0x000 No R_{LOAD} C_{LOAD}=50 pF DACC_ACR.IBCTLCHx=3	–	0.1	0.5	%. V_{VREFP}
V_{DACx_MAX}	Maximum Output Voltage on DACx	Code = 0xFFF No R_{LOAD} C_{LOAD}=50 pF DACC_ACR.IBCTLCHx=3	99.5	99.9	–	%. V_{VREFP}
FSR	Full Scale Range	Code = 0x000 to 0xFFF No R_{LOAD} C_{LOAD}=50 pF DACC_ACR.IBCTLCHx=3	99	99.8	–	%. V_{VREFP}
R_{OUT}	DAC Output Resistor	$0.3 < V_{DACx} < V_{DDIN}$ -0.3V DACC_ACR.IBCTLCHx=3 R_{LOAD}=5 KOhm	–	15	–	Ohm
		$V_{DACx} > V_{DDIN}$ -0.3V DACC_ACR.IBCTLCHx=3 R_{LOAD}=5 KOhm	–	550	–	Ohm
		$V_{DACx} < 0.3V$ DACC_ACR.IBCTLCHx=3 R_{LOAD}=5 KOhm	–	550	–	Ohm
		$V_{DACx} = V_{VREFP}/2$ DACC_ACR.IBCTLCHx = 0 (Bypass mode, buffer off) No R_{LOAD}	–	300	–	kOhm

Downloaded from Arrow.com.

56.12 Embedded Flash Characteristics

Table 56-52. Flash Characteristics

Parameter	Conditions		Min	Typ	Max	Unit
ERASE Line Assertion Time	–		230	–	–	ms
Program Cycle Time	Erase Page Mode		–	10	50	ms
	Erase Block Mode (by 8 Kbytes)		–	80	200	ms
	Erase Sector Mode		–	800	1500	ms
Full Chip Erase	512 Kbytes		–	3	6	s
	1 Mbytes		–	6	12	s
	2 Mbytes		–	13	24	s
Data Retention	At T_A=85°C, after 10K cycles(1)		10	–	–	years
	At T_A=85°C, after 1K cycles(1)		20	–	–	years
	At T_A=105°C,after 1K cycles(1)		5.5	–	–	years
Endurance	Write/Erase cycles per page, block or sector at 25°C		100K	–	–	cycles
	Write/Erase cycles per page, block or sector at 105°C		10K	–	–	cycles
Flash Active Current	Random 128-bit read at maximum frequency at 25°C	on VDDCORE = 1.2V	–	16	20	mA
		on VDDIO	–	2	10	
	Program at 25°C	on VDDCORE = 1.2V	–	2	3	
		on VDDIO	–	8	12	
	Erase at 25°C	on VDDCORE = 1.2V	–	2	2	
		on VDDIO	–	8	12	

Note: 1. Cycling over full temperature range.

Downloaded from Arrow.com.

Maximum operating frequencies are given in Table 56-53 and Table 56-54, but are limited by the Embedded Flash access time when the processor is fetching code out of it. These tables provide the device maximum operating frequency defined by the field FWS of the EEFC_FMR register. This field defines the number of wait states required to access the Embedded Flash Memory.

Table 56-53. Embedded Flash Wait States for Worst-Case Conditions

FWS	Read Operations	Maximum Operating Frequency (MHz)	
		VDDIO 1.7V	VDDIO 2.7V
0	1 cycle	21	23
1	2 cycles	42	46
2	3 cycles	63	69
3	4 cycles	84	92
4	5 cycles	106	115
5	6 cycles	125	125

Table 56-54. Embedded Flash Wait States for STH Conditions

FWS	Read Operations	Maximum Operating Frequency (MHz)	
		VDDIO 1.7V	VDDIO 2.7V
0	1 cycle	26	25
1	2 cycles	46	51
2	3 cycles	69	76
3	4 cycles	92	102
4	5 cycles	116	128
5	6 cycles	139	150
6	7 cycles	150	150

Downloaded from Arrow.com.

56.13 Timings for Worst-Case Conditions

56.13.1 AC Characteristics

56.13.1.1 Processor Clock Characteristics

Table 56-55. Processor Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Unit
$1/(t_{CPPCK})$	Processor Clock Frequency	Worst case	–	250	MHz

56.13.1.2 Master Clock Characteristics

Table 56-56. Master Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Unit
$1/(t_{CPMCK})$	Master Clock Frequency	Worst case	–	125	MHz

Downloaded from Arrow.com.

56.13.1.3 I/O Characteristics

Criteria used to define the maximum frequency of the I/Os:

- Output duty cycle (40%-60%)
- Minimum output swing: 100 mV to V_{DDIO} - 100 mV
- Addition of rising and falling time inferior to 75% of the period

Table 56-57. I/O Characteristics

Symbol	Parameter	Conditions			Min	Max	Unit
		Load	V_{DDIO}	Drive Level			
FreqMax1	Pin Group 1 (1) Maximum output frequency	10 pF	1.7V	Low	–	40	MHz
				High	–	65	
		25 pF		Low	–	20	
				High	–	33	
		10 pF	2.7V	Low	–	65	
				High	–	115	
		25 pF		Low	–	28	
				High	–	55	
$PulseminH_1$	Pin Group 1 (1) High Level Pulse Width	10 pF	1.7V	High	6.1	9.2	ns
$PulseminL_1$	Pin Group 1 (1) Low Level Pulse Width	10 pF	1.7V	High	6.1	9.2	ns
FreqMax2	Pin Group 2 (2)Maximum output frequency	10 pF	3.0V	High	–	125	MHz
				Low	–	100	
$PulseminH_2$	Pin Group 2 (2) High Level Pulse Width	10 pF	3.0V	High	3.4	4.1	ns
$PulseminL_2$	Pin Group 2 (2) Low Level Pulse Width	10 pF	3.0V	High	3.4	4.1	ns
FreqMax3	Pin Group3(3) Maximum output frequency	30 pF	3.0V	High	–	75	MHz
				Low	–	50	
$PulseminH_3$	Pin Group 3 (3) High Level Pulse Width	30 pF	3.0V	High	6.0	7.3	ns
$PulseminL_3$	Pin Group 3 (3) Low Level Pulse Width	30 pF		High	6.0	7.3	ns
FreqMax4	Pin Group 4(4)Maximum output frequency	40 pF	2.7V	–	–	51	MHz
$PulseminH_4$	Pin Group 4(4) High Level Pulse Width	40 pF	2.7V	–	7.8	11.2	ns
$PulseminL_4$	Pin Group 4(4) Low Level Pulse Width	40 pF	2.7V	–	7.8	11.2	ns

Notes:
1. Pin Group 1 = GPIO, CLOCKL
2. Pin Group 2 = GPIO_CLK
3. Pin Group 3 = GPIO_AD
4. Pin Group 4 = GPIO_MLB

Downloaded from Arrow.com.

56.13.1.4 QSPI Characteristics

Figure 56-17. QSPI Master Mode with (CPOL= NCPHA = 0) or (CPOL= NCPHA= 1)

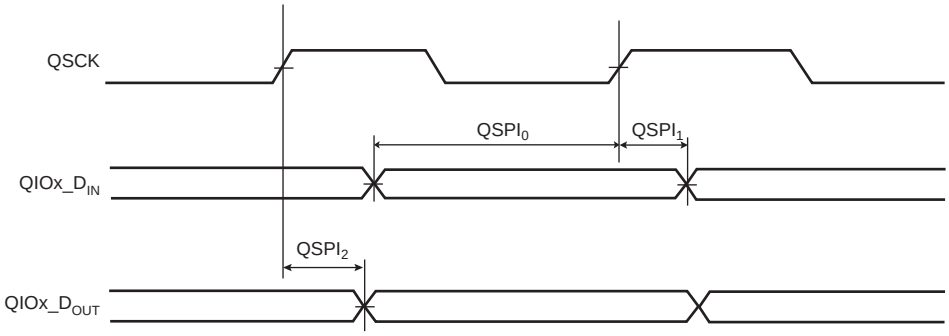
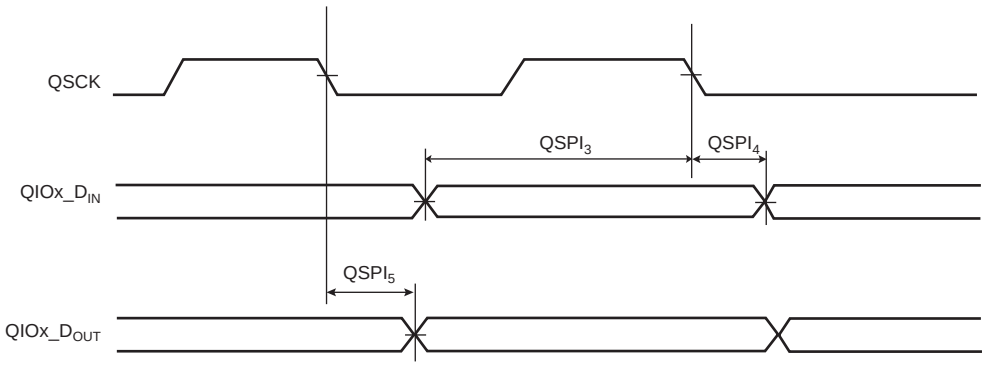


Figure 56-18. QSPI Master Mode with (CPOL = 0 and NCPHA=1) or (CPOL=1 and NCPHA= 0)



Maximum QSPI Frequency

The following formulas give maximum QSPI frequency in Master read and write modes.

Master Write Mode

The QSPI sends data to a slave device only, e.g. an LCD. The limit is given by $QSPI_2$ (or $QSPI_5$) timing. Since it gives a maximum frequency above the maximum pad speed (see Section 56.13.1.3 “I/O Characteristics”), the max QSPI frequency is the one from the pad.

Master Read Mode

$$f_{SCK}\text{max} = \frac{1}{QSPI_0(\text{or } QSPI_3) + t_{VALID}}$$

t_{VALID} is the slave time response to output data after detecting a QSCK edge.

For a non-volatile memory with t_{VALID} (or t_v) = 12 ns, f_{SCK}max = 67 MHz at V_{DDIO} = 3.3V.

Downloaded from Arrow.com.

QSPI Timings

Timings are given in the following domains:

- 1.8V domain: VDDIO from 1.7V to 1.95V, maximum external capacitor = 20 pF
- 3.3V domain: VDDIO from 2.85V to 3.6V, maximum external capacitor = 40 pF.

Table 56-58. QSPI Timings

Symbol	Parameter	Conditions	Min	Max	Unit
$QSPI_0$	QIOx data in to QSCK rising edge (input setup time)	3.3V domain	2.5	–	ns
		1.8V domain	2.9	–	ns
$QSPI_1$	QIOx data in to QSCK rising edge (input hold time)	3.3V domain	0	–	ns
		1.8V domain	0	–	ns
$QSPI_2$	QSCK rising edge to QIOx data out valid	3.3V domain	-1.3	1.9	ns
		1.8V domain	-2.5	3.0	ns
$QSPI_3$	QIOx data in to QSCK falling edge (input setup time)	3.3V domain	2.9	–	ns
		1.8V domain	3.2	–	ns
$QSPI_4$	QIOx data in to QSCK falling edge(input hold time)	3.3V domain	0	–	ns
		1.8V domain	0	–	ns
$QSPI_5$	QSCK falling edge to QIOx data out valid	3.3V domain	-1.6	1.8	ns
		1.8V domain	-2.7	3.1	ns

Downloaded from Arrow.com.

56.13.1.5 SPI Characteristics

In Figure 56-20, "SPI Master Mode with (CPOL= NCPHA = 0) or (CPOL= NCPHA= 1)" and Figure 56-21, "SPI Master Mode with (CPOL = 0 and NCPHA=1) or (CPOL=1 and NCPHA= 0)" below, the MOSI line shifting edge is represented with a hold time equal to 0. However, it is important to note that for this device, the MISO line is sampled prior to the MOSI line shifting edge. As shown in Figure 56-19, "MISO Capture in Master Mode", the device sampling point extends the propagation delay (t_p) for slave and routing delays to more than half the SPI clock period, whereas the common sampling point allows only less than half the SPI clock period.

As an example, an SPI Slave working in Mode 0 can be safely driven if the SPI Master is configured in Mode 0.

Figure 56-19. MISO Capture in Master Mode

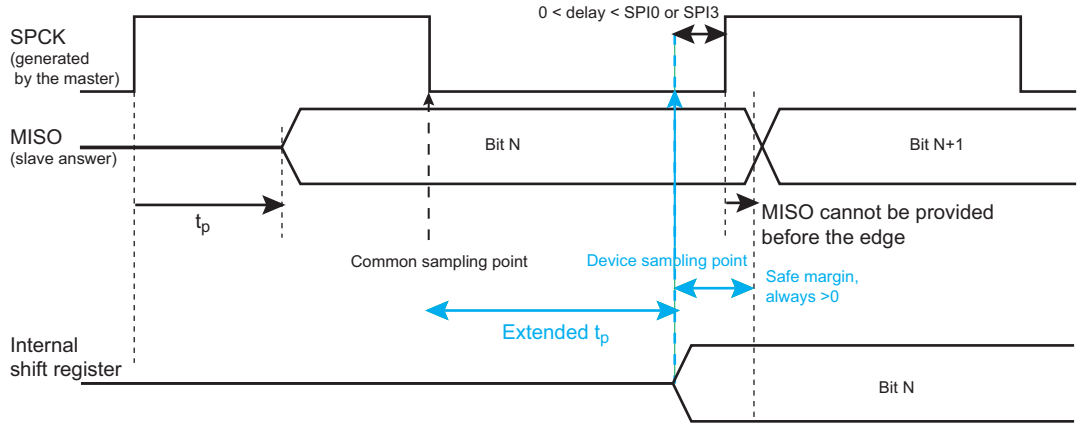


Figure 56-20. SPI Master Mode with (CPOL= NCPHA = 0) or (CPOL= NCPHA= 1)

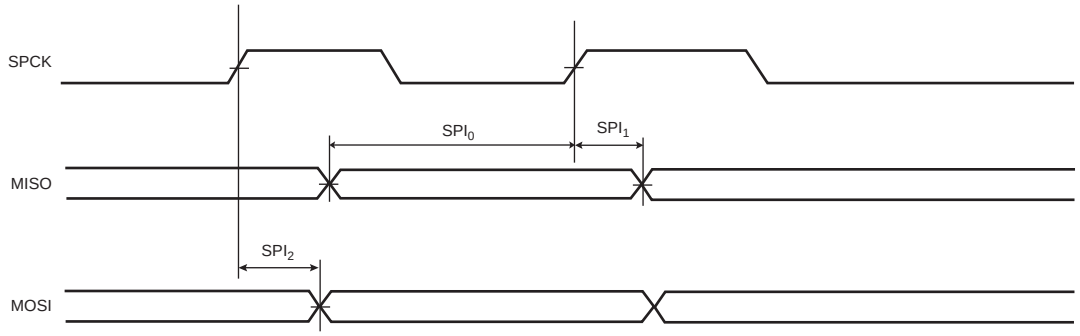
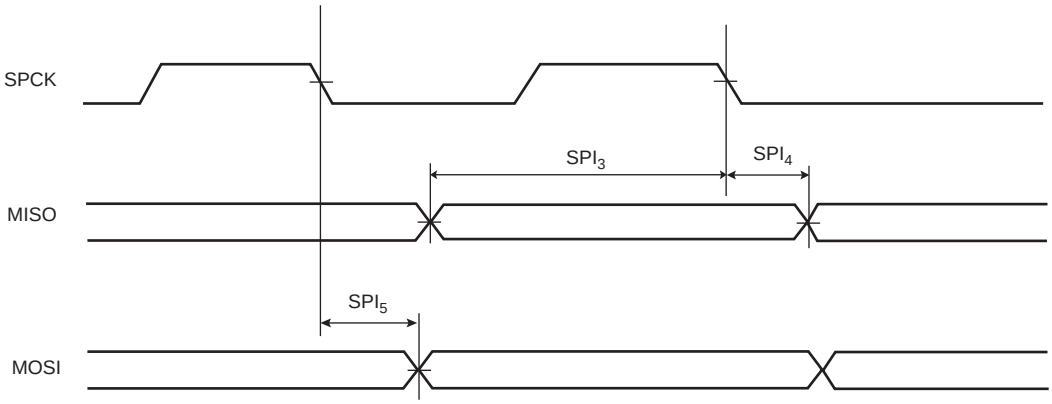


Figure 56-21. SPI Master Mode with (CPOL = 0 and NCPHA=1) or (CPOL=1 and NCPHA= 0)



Downloaded from Arrow.com.

Figure 56-22. SPI Slave Mode with (CPOL=0 and NCPHA=1) or (CPOL=1 and NCPHA=0)

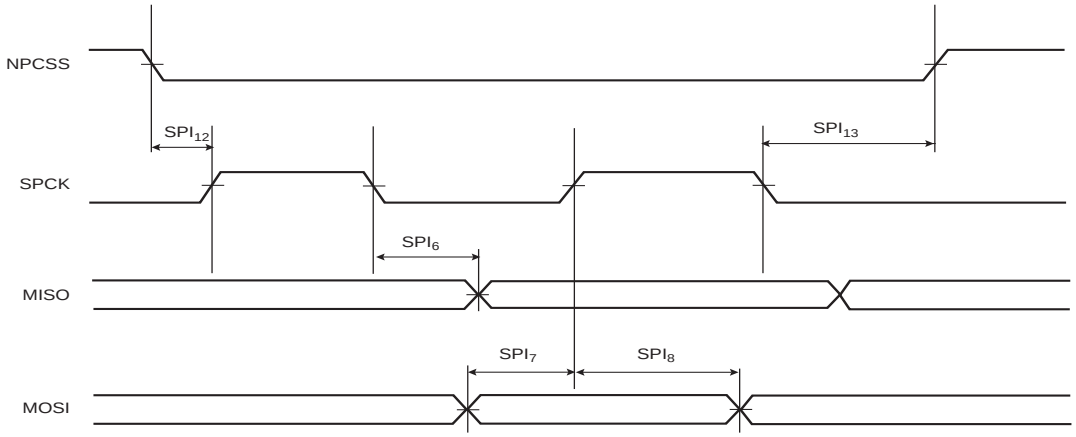
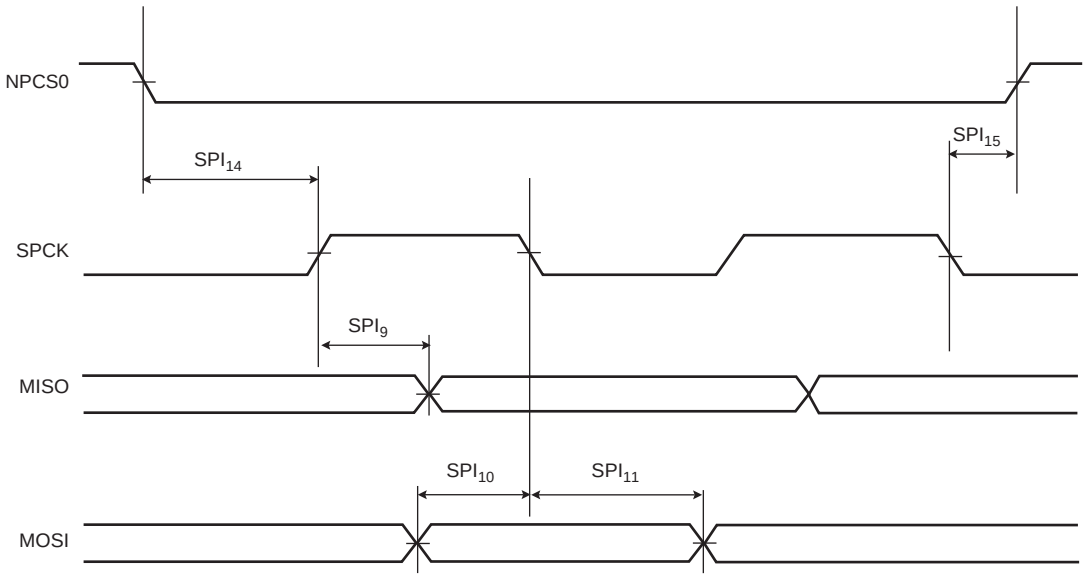


Figure 56-23. SPI Slave Mode with (CPOL = NCPHA = 0) or (CPOL= NCPHA= 1)



Maximum SPI Frequency

The following formulas give maximum SPI frequency in Master read and write modes and in Slave read and write modes.

Master Write Mode

The SPI sends data to a slave device only, e.g. an LCD. The limit is given by SPI_2 (or SPI_5) timing. Since it gives a maximum frequency above the maximum pad speed (see Section 56.13.1.3 “I/O Characteristics”), the max SPI frequency is the one from the pad.

Master Read Mode

$$f_{SPCK}max = \frac{1}{SPI_0(\text{or } SPI_3) + t_{valid}}$$

t_{valid} is the slave time response to output data after detecting an SPCK edge.

For a non-volatile memory with t_{VALID} (or t_v) = 5 ns, $f_{SPCK}max$ = 57 MHz at V_{DDIO} = 3.3V.

Downloaded from Arrow.com.

Slave Read Mode

In slave mode, SPCK is the input clock for the SPI. The max SPCK frequency is given by setup and hold timings SPI_7/SPI_8(or SPI_{10}/SPI_{11}). Since this gives a frequency well above the pad limit, the limit in slave read mode is given by SPCK pad.

Slave Write Mode

$$f_{SPCK}max = \frac{1}{2x(SPI_{6max}(\text{or } SPI_{9max}) + t_{setup})}$$

t_{setup} is the setup time from the master before sampling data.

Downloaded from Arrow.com.

SPI Timings

Timings are given in the following domains:

- 1.8V domain: V_{DDIO} from 1.7V to 1.95V, maximum external capacitor = 20 pF
- 3.3V domain: V_{DDIO} from 2.85V to 3.6V, maximum external capacitor = 40 pF

Table 56-59. SPI Timings

Symbol	Parameter	Conditions	Min	Max	Unit
SPI_0	MISO Setup time before SPCK rises (master)	3.3V domain	12.4	–	ns
		1.8V domain	14.6	–	ns
SPI_1	MISO Hold time after SPCK rises (master)	3.3V domain	0	–	ns
		1.8V domain	0	–	ns
SPI_2	SPCK rising to MOSI Delay (master)	3.3V domain	-3.7	2.2	ns
		1.8V domain	-3.8	2.7	ns
SPI_3	MISO Setup time before SPCK falls (master)	3.3V domain	12.6	–	ns
		1.8V domain	15.13	–	ns
SPI_4	MISO Hold time after SPCK falls (master)	3.3V domain	0	–	ns
		1.8V domain	0	–	ns
SPI_5	SPCK falling to MOSI Delay (master)	3.3V domain	-3.6	2.0	ns
		1.8V domain	-3.3	2.8	ns
SPI_6	SPCK falling to MISO Delay (slave)	3.3V domain	3.0	11.9	ns
		1.8V domain	3.5	13.9	ns
SPI_7	MOSI Setup time before SPCK rises (slave)	3.3V domain	1.2	–	ns
		1.8V domain	1.5	–	ns
SPI_8	MOSI Hold time after SPCK rises (slave)	3.3V domain	0.6	–	ns
		1.8V domain	0.8	–	ns
SPI_9	SPCK rising to MISO Delay (slave)	3.3V domain	3.0	12.0	ns
		1.8V domain	3.4	13.7	ns
SPI_{10}	MOSI Setup time before SPCK falls (slave)	3.3V domain	1.2	–	ns
		1.8V domain	1.5	–	ns
SPI_{11}	MOSI Hold time after SPCK falls (slave)	3.3V domain	0.6	–	ns
		1.8V domain	0.8	–	ns
SPI_{12}	NPCS setup to SPCK rising (slave)	3.3V domain	3.9	–	ns
		1.8V domain	4.4	–	ns
SPI_{13}	NPCS hold after SPCK falling (slave)	3.3V domain	0	–	ns
		1.8V domain	0	–	ns
SPI_{14}	NPCS setup to SPCK falling (slave)	3.3V domain	4.0	–	ns
		1.8V domain	4.1	–	ns
SPI_{15}	NPCS hold after SPCK falling (slave)	3.3V domain	0	–	ns
		1.8V domain	0	–	ns

Note that in SPI master mode, the device does not sample the data (MISO) on the opposite edge where the data clocks out (MOSI), but the same edge is used. See Figure 56-20 and Figure 56-21.

Downloaded from Arrow.com.

56.13.1.6 HSMCI Timings

The High-speed MultiMedia Card Interface (HSMCI) supports the MultiMedia Card (MMC) Specification V4.3, the SD Memory Card Specification V2.0, the SDIO V2.0 specification and CE-ATA V1.1.

56.13.1.7 SDRAM Timings

The SDRAM Controller satisfies the timings of standard SDR-133 and LP-SDR-133 modules. SDR-133 and LP-SDR-133 timings are specified by the JEDEC standard.

Downloaded from Arrow.com.

56.13.1.8 SMC Timings

Timings are given in the following domains:

- 1.8V domain: VDDIO from 1.7V to 1.95V, maximum external capacitor = 30 pF
- 3.3V domain: VDDIO from 2.85V to 3.6V, maximum external capacitor = 50 pF

Timings are given assuming a capacitance load on data, control and address pads.

In the tables that follow, t_{CPMCK} is MCK period.

Read Timings

Table 56-60. SMC Read Signals - NRD Controlled (READ_MODE = 1)

Symbol	VDDIO Supply / Parameter	1.8V Domain (Min)	3.3V Domain (Min)	1.8V Domain (Max)	3.3V Domain (Max)	Unit
NO HOLD Settings (NRD_HOLD = 0)						
SMC_1	Data Setup before NRD High	17.2	14.3	–	–	ns
SMC_2	Data Hold after NRD High	0	0	–	–	ns
HOLD Settings (NRD_HOLD ≠ 0)						
SMC_3	Data Setup before NRD High	15.2	12.1	–	–	ns
SMC_4	Data Hold after NRD High	0	0	–	–	ns
HOLD or NO HOLD Settings (NRD_HOLD ≠ 0, NRD_HOLD = 0)						
SMC_5	A0–A22 Valid before NRD High	(NRD_SETUP + NRD_PULSE) × t_{CPMCK} - 5.1	(NRD_SETUP + NRD_PULSE) × t_{CPMCK} - 4.3	–	–	ns
SMC_6	NCS low before NRD High	(NRD_SETUP + NRD_PULSE - NCS_RD_SETUP) × t_{CPMCK} - 3.5	(NRD_SETUP + NRD_PULSE - NCS_RD_SETUP) × t_{CPMCK} - 2.4	–	–	ns
SMC_7	NRD Pulse Width	NRD_PULSE × t_{CPMCK} - 0.7	NRD_PULSE × t_{CPMCK} - 0.3	–	–	ns

Table 56-61. SMC Read Signals - NCS Controlled (READ_MODE = 0)

Symbol	VDDIO Supply / Parameter	1.8V Domain (Min)	3.3V Domain (Min)	1.8V Domain (Max)	3.3V Domain (Max)	Unit
NO HOLD Settings (NCS_RD_HOLD = 0)						
SMC_8	Data Setup before NCS High	24.9	21.4	–	–	ns
SMC_9	Data Hold after NCS High	0	0	–	–	ns
HOLD Settings (NCS_RD_HOLD ≠ 0)						
SMC_{10}	Data Setup before NCS High	13.4	11.7	–	–	ns
SMC_{11}	Data Hold after NCS High	0	0	–	–	ns
HOLD or NO HOLD Settings (NCS_RD_HOLD ≠ 0, NCS_RD_HOLD = 0)						
SMC_{12}	A0–A22 valid before NCS High	(NCS_RD_SETUP + NCS_RD_PULSE) × t_{CPMCK} - 4.0	(NCS_RD_SETUP + NCS_RD_PULSE) × t_{CPMCK} - 3.9	–	–	ns
SMC_{13}	NRD low before NCS High	(NCS_RD_SETUP + NCS_RD_PULSE - NRD_SETUP) × t_{CPMCK} - 2.8	(NCS_RD_SETUP + NCS_RD_PULSE - NRD_SETUP) × t_{CPMCK} - 4.2	–	–	ns
SMC_{14}	NCS Pulse Width	NCS_RD_PULSE length × t_{CPMCK} - 0.9	NCS_RD_PULSE length × t_{CPMCK} - 0.2	–	–	ns

Downloaded from Arrow.com.

Table 56-62. SMC Write Signals - NWE Controlled (WRITE_MODE = 1)

Symbol	Parameter	VDDIO Supply 1.8V Domain Min	VDDIO Supply 3.3V Domain Min	1.8V Domain Max	3.3V Domain Max	Unit
HOLD or NO HOLD Settings (NWE_HOLD ≠ 0, NWE_HOLD = 0)						
SMC_{15}	Data Out Valid before NWE High	NWE_PULSE × t_{CPMCK} - 5.4	NWE_PULSE × t_{CPMCK} - 4.6	–	–	ns
SMC_{16}	NWE Pulse Width	NWE_PULSE × t_{CPMCK} - 0.7	NWE_PULSE × t_{CPMCK} - 0.3	–	–	ns
SMC_{17}	A0–A22 valid before NWE low	NWE_SETUP × t_{CPMCK} - 4.9	NWE_SETUP × t_{CPMCK} - 4.2	–	–	ns
SMC_{18}	NCS low before NWE high	(NWE_SETUP - NCS_RD_SETUP + NWE_PULSE) × t_{CPMCK} - 3.2	(NWE_SETUP - NCS_RD_SETUP + NWE_PULSE) × t_{CPMCK} - 2.2	–	–	ns
HOLD Settings (NWE_HOLD ≠ 0)						
SMC_{19}	NWE High to Data OUT, NBS0/A0 NBS1, A1, A2–A25 change	NWE_HOLD × t_{CPMCK} - 4.6	NWE_HOLD × t_{CPMCK} - 3.9	–	–	ns
SMC_{20}	NWE High to NCS Inactive (1)	(NWE_HOLD - NCS_WR_HOLD) × t_{CPMCK} - 3.9	(NWE_HOLD - NCS_WR_HOLD) × t_{CPMCK} - 3.6	–	–	ns
NO HOLD Settings (NWE_HOLD = 0)						
SMC_{21}	NWE High to Data OUT, NBS0/A0 NBS1, A1, A2–A25, NCS change(1)	2.1	1.5	–	–	ns

Notes: 1. Hold length = total cycle duration - setup duration - pulse duration. “hold length” is for “NCS_WR_HOLD length” or “NWE_HOLD length”

Table 56-63. SMC Write NCS Controlled (WRITE_MODE = 0)

Symbol	Parameter	VDDIO Supply 1.8V Domain Min	VDDIO Supply 3.3V Domain Min	1.8V Domain Max	3.3V Domain Max	Unit
SMC_{22}	Data Out Valid before NCS High	NCS_WR_PULSE × t_{CPMCK} - 2.8	NCS_WR_PULSE × t_{CPMCK} - 3.9	—	—	ns
SMC_{23}	NCS Pulse Width	NCS_WR_PULSE × t_{CPMCK} - 0.9	NCS_WR_PULSE × t_{CPMCK} - 0.2	—	—	ns
SMC_{24}	A0–A22 valid before NCS low	NCS_WR_SETUP × t_{CPMCK} - 4.0	NCS_WR_SETUP × t_{CPMCK} - 4.6	—	—	ns
SMC_{25}	NWE low before NCS high	(NCS_WR_SETUP - NWE_SETUP + NCS pulse) × t_{CPMCK} - 4.6	(NCS_WR_SETUP - NWE_SETUP + NCS pulse) × t_{CPMCK} - 4.6	—	—	ns
SMC_{26}	NCS High to Data Out, A0–A25, change	NCS_WR_HOLD × t_{CPMCK} - 4.4	NCS_WR_HOLD × t_{CPMCK} - 3.4	—	—	ns
SMC_{27}	NCS High to NWE Inactive	(NCS_WR_HOLD - NWE_HOLD) × t_{CPMCK} - 2.8	(NCS_WR_HOLD - NWE_HOLD) × t_{CPMCK} - 2.4	—	—	ns

Downloaded from Arrow.com.

Figure 56-24. SMC Timings - NCS Controlled Read and Write

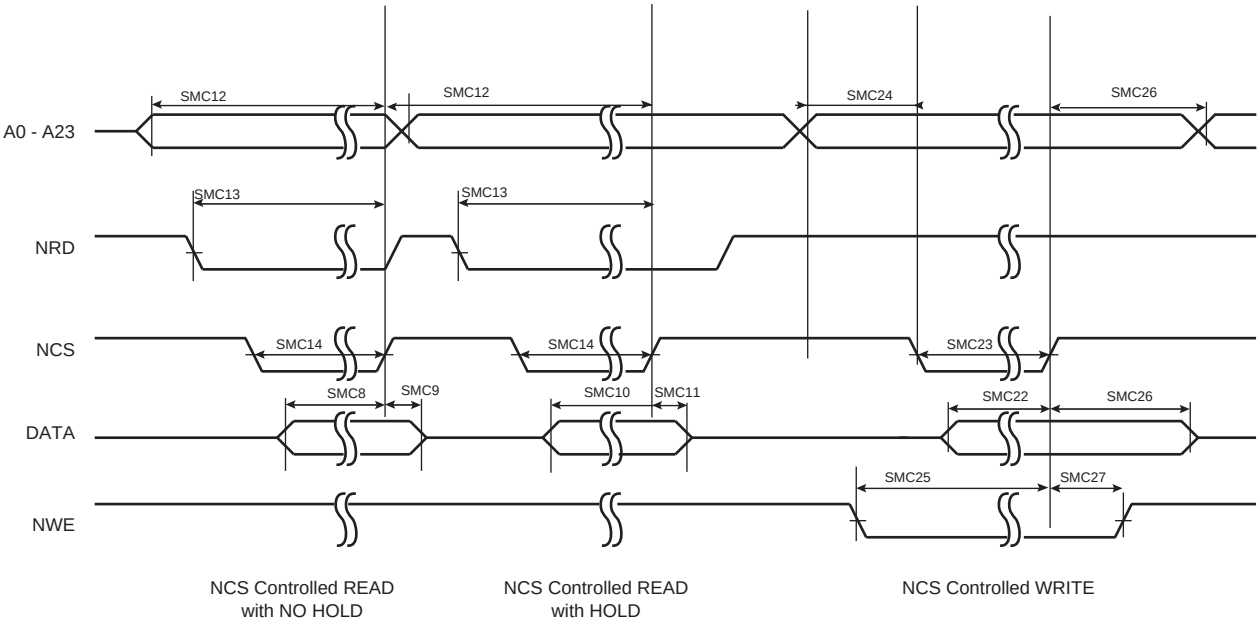
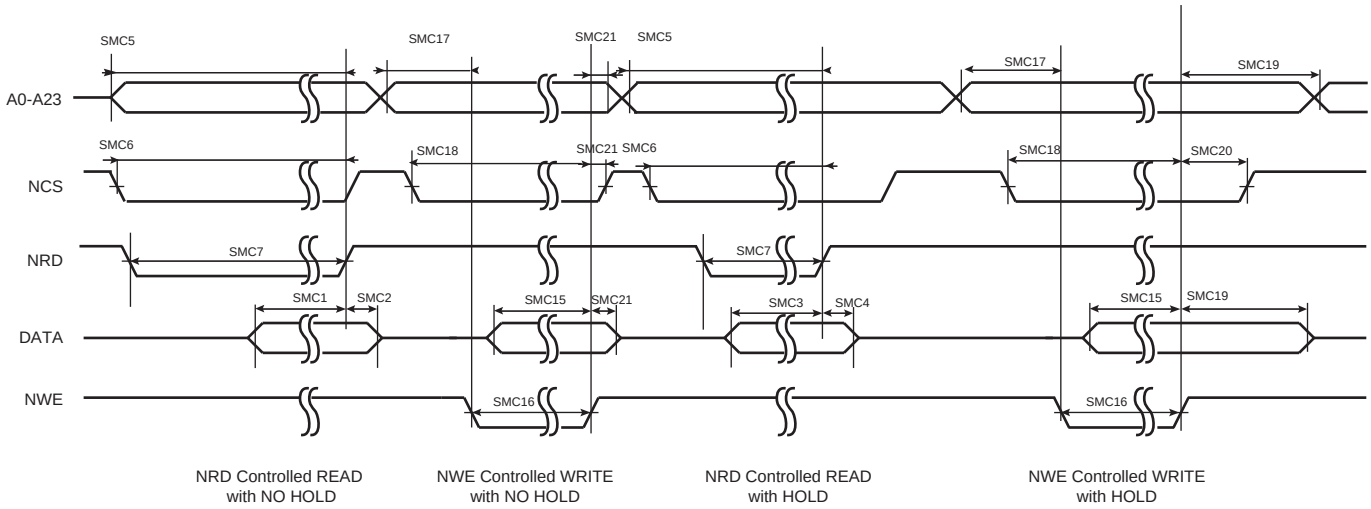


Figure 56-25. SMC Timings - NRD Controlled Read and NWE Controlled Write



Downloaded from Arrow.com.

56.13.1.9 USART in SPI Mode Timings

Figure 56-26. USART SPI Master Mode

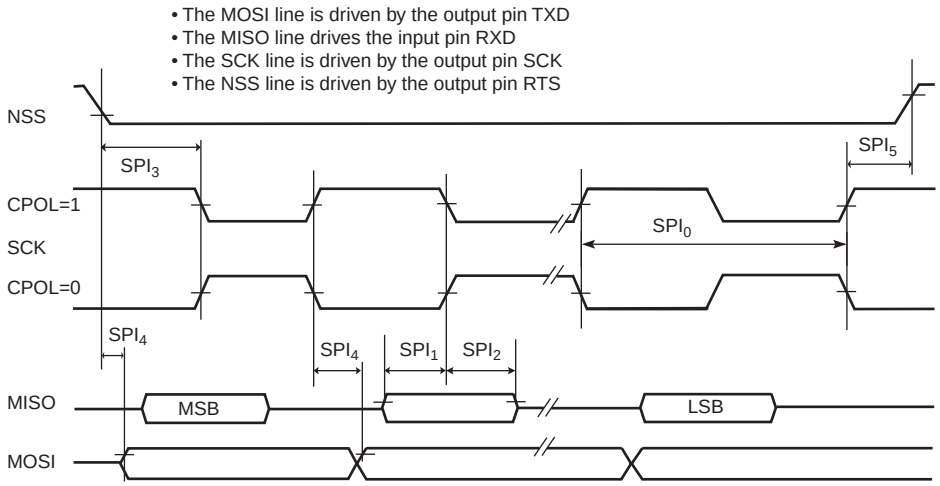
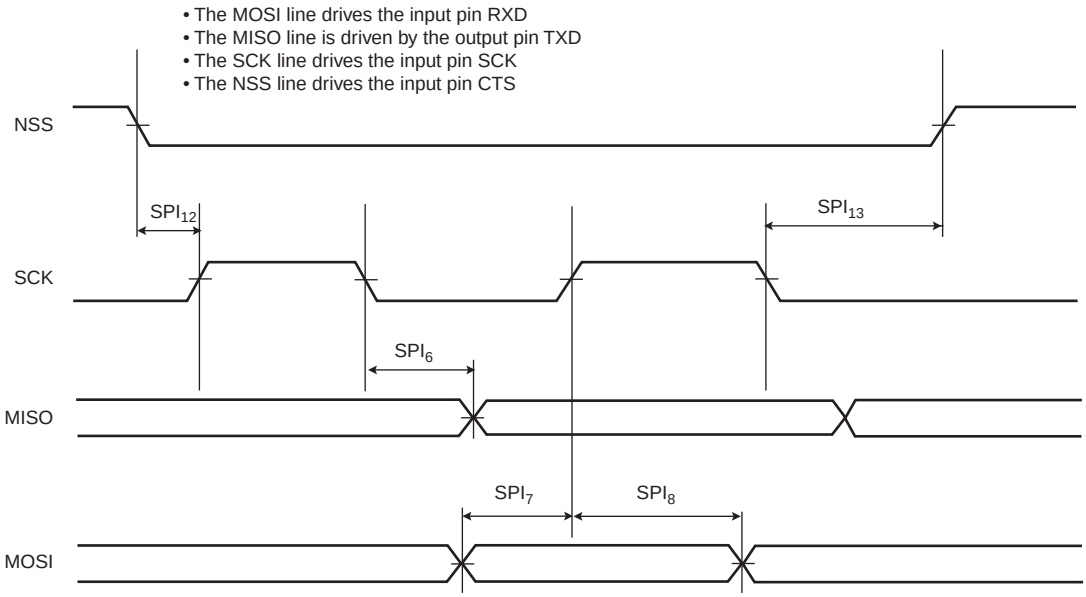
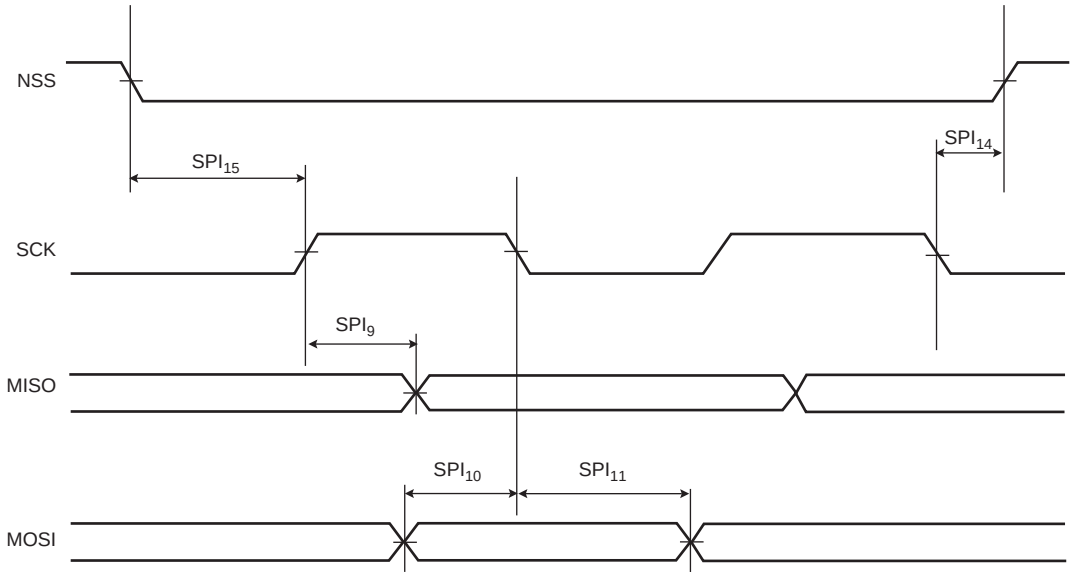


Figure 56-27. USART SPI Slave Mode (Mode 1 or 2)



Downloaded from Arrow.com.

Figure 56-28. USART SPI Slave Mode (Mode 0 or 3)



Downloaded from Arrow.com.

USART SPI TImings

Timings are given in the following domains:

- 1.8V domain: VDDIO from 1.7V to 1.95V, maximum external capacitor = 20 pF
- 3.3V domain: VDDIO from 2.85V to 3.6V, maximum external capacitor = 40 pF

Table 56-64. USART SPI Timings

Symbol	Parameter	Conditions	Min	Max	Unit
Master Mode					
SPI_0	SCK Period	1.8V domain 3.3V domain	MCK/6	–	ns
SPI_1	Input Data Setup Time	1.8V domain 3.3V domain	2.8 2.5	–	ns
SPI_2	Input Data Hold Time	1.8V domain 3.3V domain	0.5 0.2	–	ns
SPI_3	Chip Select Active to Serial Clock	1.8V domain 3.3V domain	-1.1 -0.9	–	ns
SPI_4	Output Data Setup Time	1.8V domain 3.3V domain	-1.9 -1.9	10.9 10.4	ns
SPI_5	Serial Clock to Chip Select Inactive	1.8V domain 3.3V domain	-2.4 -2.4	-1.9 -1.9	ns
Slave Mode					
SPI_6	SCK falling to MISO	1.8V domain 3.3V domain	3.6 2.9	16.8 13.9	ns
SPI_7	MOSI Setup time before SCK rises	1.8V domain 3.3V domain	2.4 2.0	–	ns
SPI_8	MOSI Hold time after SCK rises	1.8V domain 3.3V domain	0.4 0.2	–	ns
SPI_9	SCK rising to MISO	1.8V domain 3.3V domain	3.5 3.0	16.2 13.5	ns
SPI_{10}	MOSI Setup time before SCK falls	1.8V domain 3.3V domain	2.2 2.1	–	ns
SPI_{11}	MOSI Hold time after SCK falls	1.8V domain 3.3V domain	0.6 0.4	–	ns
SPI_{12}	NPCS0 setup to SCK rising	1.8V domain 3.3V domain	1.6 0.6	–	ns
SPI_{13}	NPCS0 hold after SCK falling	1.8V domain 3.3V domain	1.1 0.6	–	ns
SPI_{14}	NPCS0 setup to SCK falling	1.8V domain 3.3V domain	1.3 0.6	–	ns
SPI_{15}	NPCS0 hold after SCK rising	1.8V domain 3.3V domain	0.9 0.7	–	ns

Downloaded from Arrow.com.

56.13.1.10 Two-wire Serial Interface Characteristics

Table 56-65 describes the requirements for devices connected to the Two-wire Serial Bus.

For timing symbols, refer to Figure 56-29.

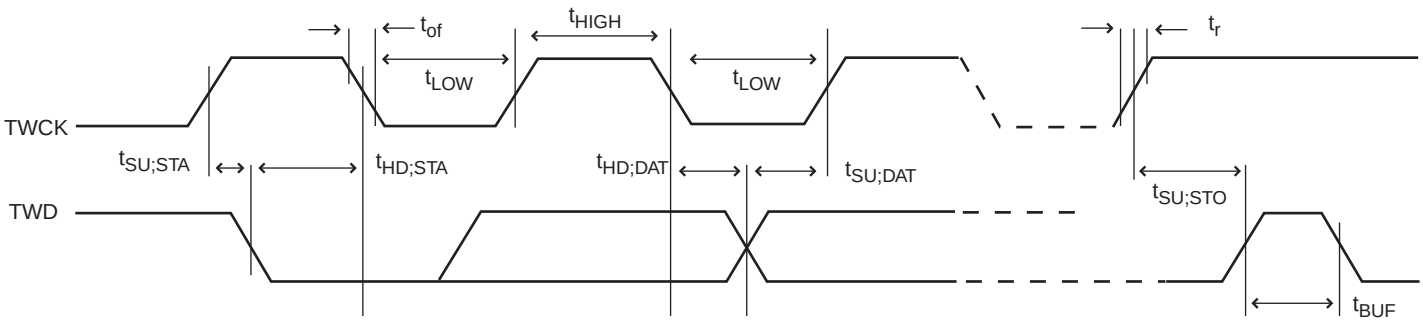
Table 56-65. Two-wire Serial Bus Requirements

Symbol	Parameter	Condition	Min	Max	Unit
V_{IL}	Low-level Input Voltage	–	-0.3	0.3 V_{DDIO}	V
V_{IH}	High-level Input Voltage	–	0.7 × V_{DDIO}	V_{CC} + 0.3	V
V_{hys}	Hysteresis of Schmitt Trigger Inputs	–	0.150	–	V
V_{OL}	Low-level Output Voltage	3 mA sink current	–	0.4	V
t_R	Rise Time for both TWD and TWCK		20 + 0.1C_b (1)(2)	300	ns
t_{OF}	Output Fall Time from V_{IHmin} to V_{ILmax}	10 pF < C_b < 400 pF Figure 56-29	20 + 0.1C_b (1)(2)	250	ns
C_i (1)	Capacitance for each I/O Pin	–	–	10	pF
f_{TWCK}	TWCK Clock Frequency	–	0	400	kHz
R_P	Value of Pull-up resistor	f_{TWCK} ≤ 100 kHz	(V_{DDIO} - 0.4V) ÷ 3mA	1000ns ÷ C_b	Ω
		f_{TWCK} > 100 kHz		300ns ÷ C_b	Ω
t_{LOW}	Low Period of the TWCK clock	f_{TWCK} ≤ 100 kHz	(3)	–	µs
		f_{TWCK} > 100 kHz	(3)	–	µs
t_{HIGH}	High period of the TWCK clock	f_{TWCK} ≤ 100 kHz	(4)	–	µs
		f_{TWCK} > 100 kHz	(4)	–	µs
$t_{HD;STA}$	Hold Time (repeated) START Condition	f_{TWCK} ≤ 100 kHz	t_{HIGH}	–	µs
		f_{TWCK} > 100 kHz	t_{HIGH}	–	µs
$t_{SU;STA}$	Set-up time for a repeated START condition	f_{TWCK} ≤ 100 kHz	t_{HIGH}	–	µs
		f_{TWCK} > 100 kHz	t_{HIGH}	–	µs
$t_{HD;DAT}$	Data hold time	f_{TWCK} ≤ 100 kHz	0	3 × t_{CPMCK} (5)	µs
		f_{TWCK} > 100 kHz	0	3 ×t_{CPMCK} (5)	µs
$t_{SU;DAT}$	Data setup time	f_{TWCK} ≤ 100 kHz	t_{LOW} - 3 × t_{CPMCK} (5)	–	ns
		f_{TWCK} > 100 kHz	t_{LOW} - 3 × t_{CPMCK} (5)	–	ns
$t_{SU;STO}$	Setup time for STOP condition	f_{TWCK} ≤ 100 kHz	t_{HIGH}	–	µs
		f_{TWCK} > 100 kHz	t_{HIGH}	–	µs
$t_{HD;STA}$	Hold Time (repeated) START Condition	f_{TWCK} ≤ 100 kHz	t_{HIGH}	–	µs
		f_{TWCK} > 100 kHz	t_{HIGH}	–	µs

Notes:
1. Required only for f_{TWCK} > 100 kHz.
2. C_b = capacitance of one bus line in pF. Per I^2C standard, C_b max = 400pF.
3. The TWCK low period is defined as follows: t_{LOW} = ((CLDIV × 2^{CKDIV}) + 4) × t_{MCK}
4. The TWCK high period is defined as follows: t_{HIGH} = ((CHDIV × 2^{CKDIV}) + 4) × t_{MCK}
5. t_{CPMCK} = MCK bus period

Downloaded from Arrow.com.

Figure 56-29. Two-wire Serial Bus Timing



Downloaded from Arrow.com.

56.13.1.11 GMAC Characteristics

Timing Conditions

Table 56-66. Load Capacitance on Data, Clock Pads

Supply	C_L	
	Max	Min
3.3V	20 pF	0 pF

Timing Constraints

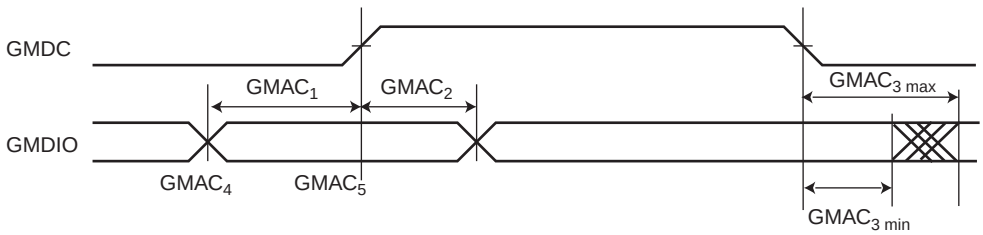
The GMAC must be constrained so as to satisfy the timings of standards given in Table 56-67 and Table 56-68, in MAX corner**.**

Table 56-67. GMAC Signals Relative to GMDC

Symbol	Parameter	Min	Max	Unit
$GMAC_1$	Setup for GMDIO from GMDC rising	10	–	ns
$GMAC_2$	Hold for GMDIO from GMDC rising	10	–	
$GMAC_3$	GMDIO toggling from GMDC falling	0(1)	10(1)	

Note: 1. For GMAC output signals, min and max access time are defined. The min access time is the time between the GMDC falling edge and the signal change. The max access timing is the time between the GMDC falling edge and the signal stabilizes. Figure 56-30 illustrates min and max accesses for $GMAC_3$.

Figure 56-30. Min and Max Access Time of GMAC Output Signals



Downloaded from Arrow.com.

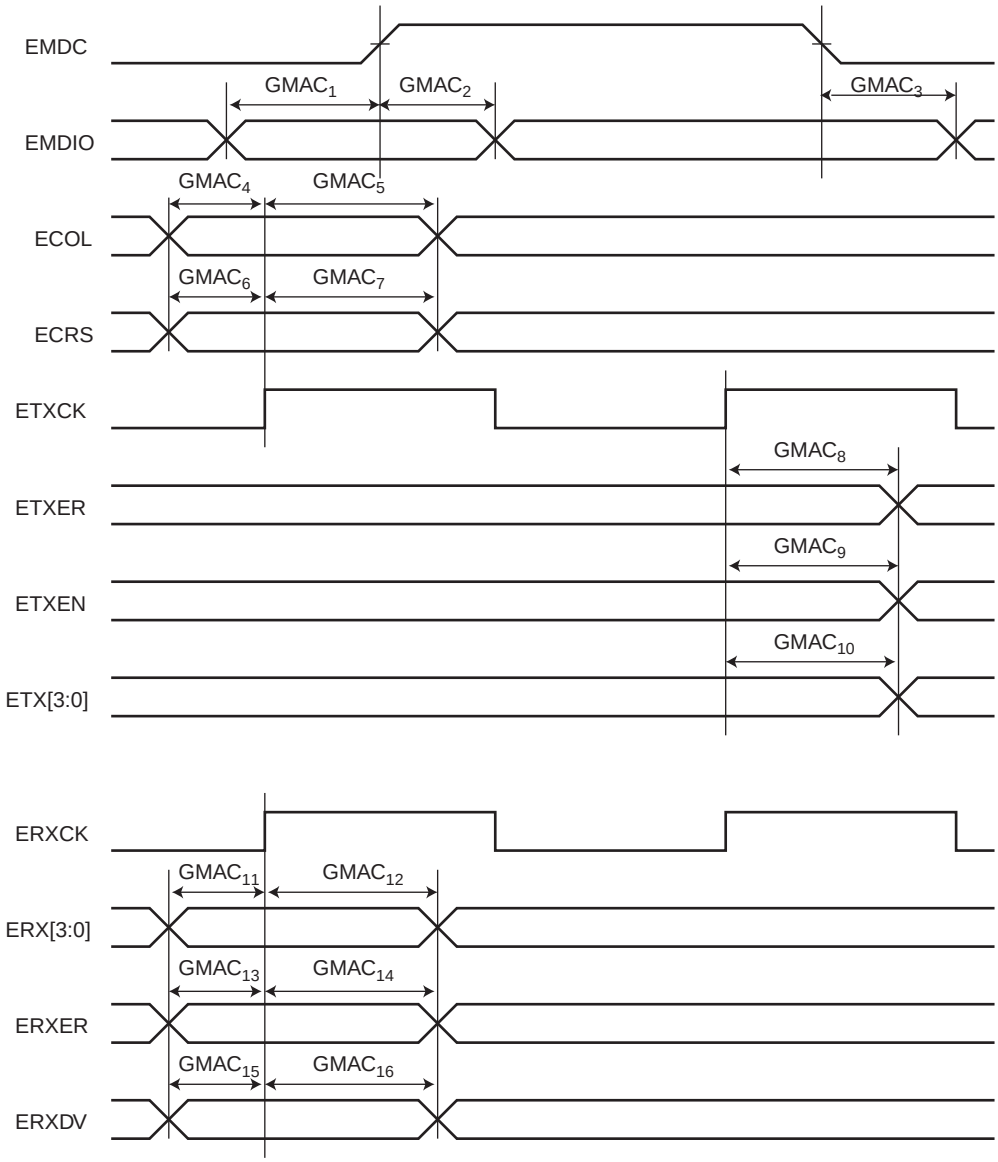
MII Mode

Table 56-68. GMAC MII Mode Timings

Symbol	Parameter	Min	Max	Unit
$GMAC_4$	Setup for GCOL from GTXCK rising	10	–	ns
$GMAC_5$	Hold for GCOL from GTXCK rising	10	–	
$GMAC_6$	Setup for GCRS from GTXCK rising	10	–	
$GMAC_7$	Hold for GCRS from GTXCK rising	10	–	
$GMAC_8$	GTXER toggling from GTXCK rising	10	25	
$GMAC_9$	GTXEN toggling from GTXCK rising	10	25	
$GMAC_{10}$	GTX toggling from GTXCK rising	10	25	
$GMAC_{11}$	Setup for GRX from GRXCK	10	–	
$GMAC_{12}$	Hold for GRX from GRXCK	10	–	
$GMAC_{13}$	Setup for GRXER from GRXCK	10	–	
$GMAC_{14}$	Hold for GRXER from GRXCK	10	–	
$GMAC_{15}$	Setup for GRXDV from GRXCK	10	–	
$GMAC_{16}$	Hold for GRXDV from GRXCK	10	–	

Downloaded from Arrow.com.

Figure 56-31. GMAC MII Mode Signals



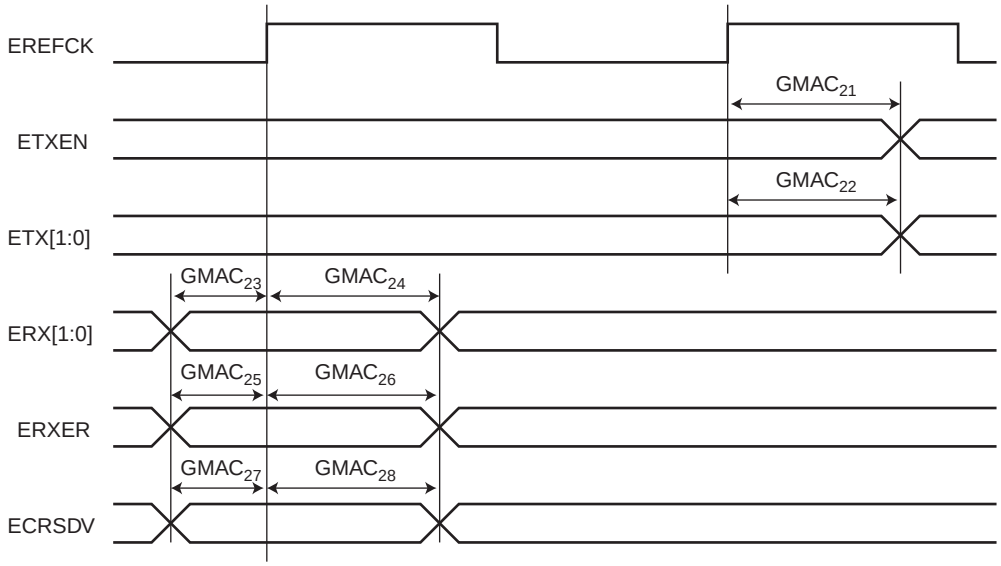
Downloaded from Arrow.com.

RMII Mode

Table 56-69. GMAC RMII Mode Timings

Symbol	Parameter	Min	Max	Unit
$GMAC_{21}$	ETXEN toggling from EREFCK rising	2	16	ns
$GMAC_{22}$	ETX toggling from EREFCK rising	2	16	
$GMAC_{23}$	Setup for ERX from EREFCK rising	4	–	
$GMAC_{24}$	Hold for ERX from EREFCK rising	2	–	
$GMAC_{25}$	Setup for ERXER from EREFCK rising	4	–	
$GMAC_{26}$	Hold for ERXER from EREFCK rising	2	–	
$GMAC_{27}$	Setup for ECRSDV from EREFCK rising	4	–	
$GMAC_{28}$	Hold for ECRSDV from EREFCK rising	2	–	

Figure 56-32. GMAC RMII Mode Signals



Downloaded from Arrow.com.

56.13.1.12 SSC Timings

Timing Conditions

Timings are given assuming the load capacitance in Table 56-70.

Table 56-70. Load Capacitance

Supply	C_L Max
3.3V	30 pF
1.8V	20 pF

Timing Extraction

Figure 56-33. SSC Transmitter, TK and TF in Output

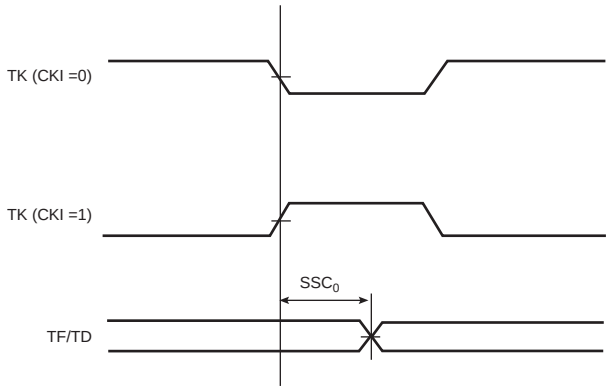
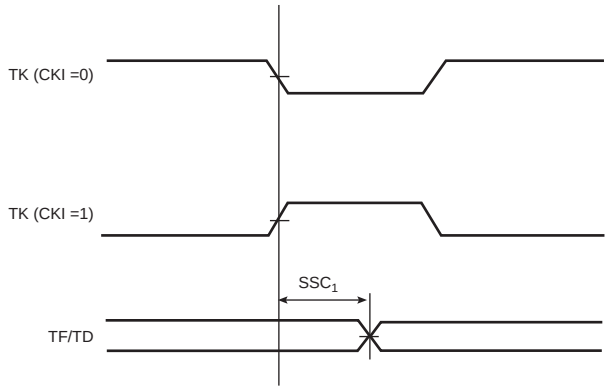


Figure 56-34. SSC Transmitter, TK in Input and TF in Output



Downloaded from Arrow.com.

Figure 56-35. SSC Transmitter, TK in Output and TF in Input

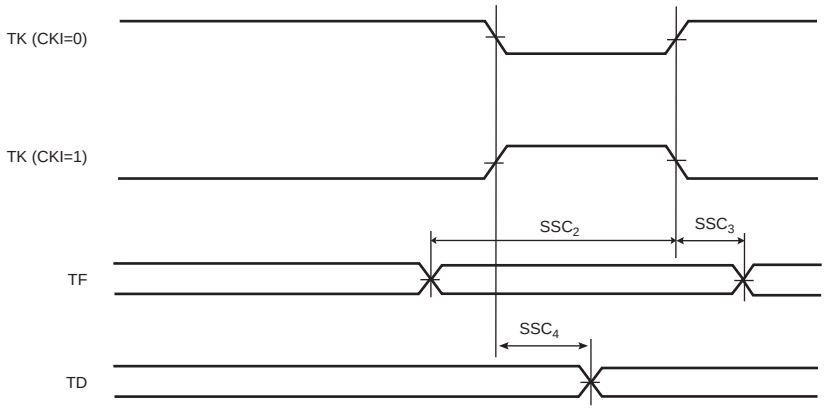


Figure 56-36. SSC Transmitter, TK and TF in Input

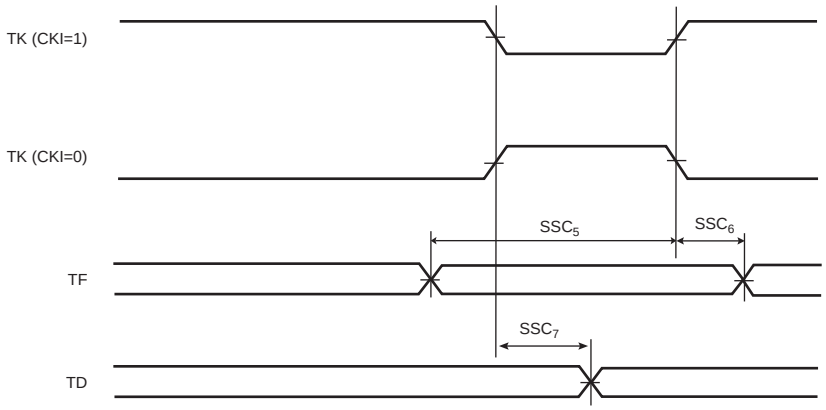
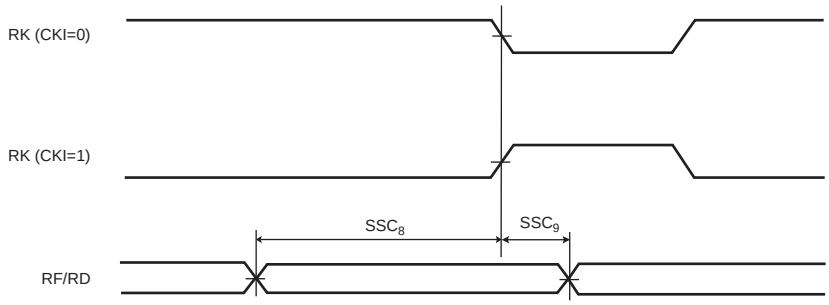


Figure 56-37. SSC Receiver RK and RF in Input



Downloaded from Arrow.com.

Figure 56-38. SSC Receiver, RK in Input and RF in Output

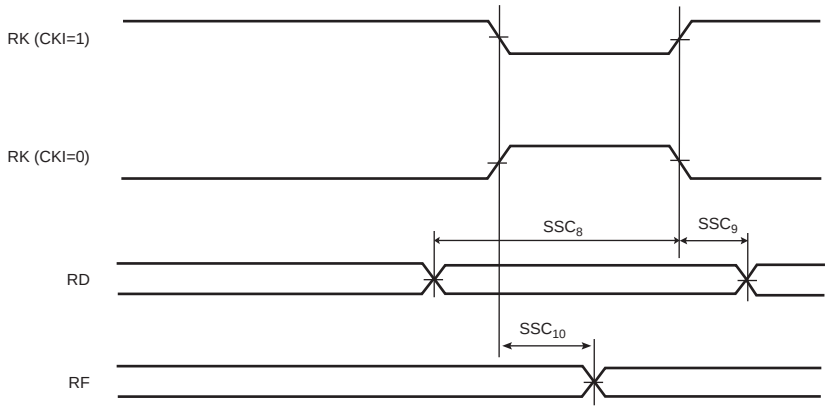


Figure 56-39. SSC Receiver, RK and RF in Output

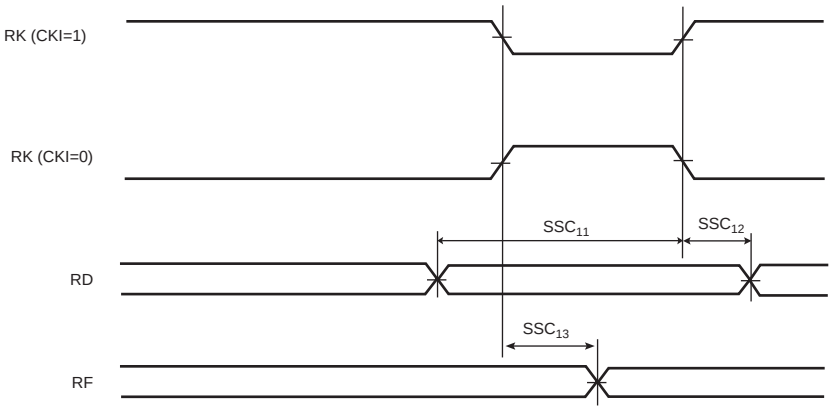
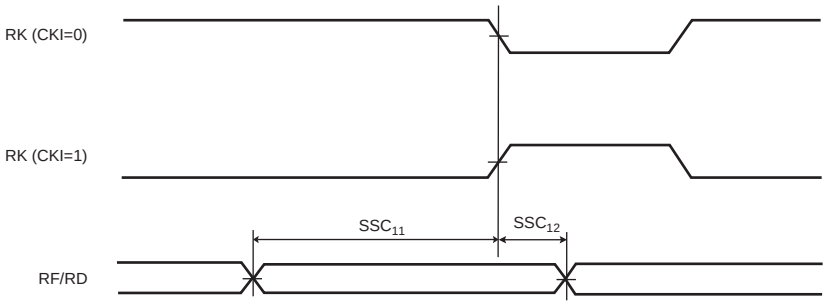


Figure 56-40. SSC Receiver, RK in Output and RF in Input



Downloaded from Arrow.com.

Table 56-71. SSC Timings with 3.3V Peripheral Supply

Symbol	Parameter	Condition	Min	Max	Unit
Transmitter					
SSC_0	TK edge to TF/TD (TK output, TF output)	–	-3.9(1)	4.0 (1)	ns
SSC_1	TK edge to TF/TD (TK input, TF output)	–	3.1(1)	12.7(1)	ns
SSC_2	TF setup time before TK edge (TK output)	–	13.6	–	ns
SSC_3	TF hold time after TK edge (TK output)	–	0	–	ns
SSC_4	TK edge to TF/TD (TK output, TF input)	–	-3.9(1)	3.0(1)	ns
		STTDLY = 0 START = 4, 5 or 7	-3.9 + (2× t_{CPMCK})(1)	3.0 + (2 × t_{CPMCK})(1)	
SSC_5	TF setup time before TK edge (TK input)	–	0	–	ns
SSC_6	TF hold time after TK edge (TK input)	–	t_{CPMCK}	–	ns
SSC_7	TK edge to TF/TD (TK input, TF input)	–	3.1(1)	11.8(1)	ns
		STTDLY = 0 START = 4, 5 or 7	3.1 + (3 × t_{CPMCK})(1)	11.8 + (3 × t_{CPMCK})(1)	
Receiver					
SSC_8	RF/RD setup time before RK edge (RK input)	–	0	–	ns
SSC_9	RF/RD hold time after RK edge (RK input)	–	t_{CPMCK}	–	ns
SSC_{10}	RK edge to RF (RK input)	–	2.9(1)	9.2(1)	ns
SSC_{11}	RF/RD setup time before RK edge (RK output)	–	10.1 - t_{CPMCK}	–	ns
SSC_{12}	RF/RD hold time after RK edge (RK output)	–	t_{CPMCK} - 2.8	–	ns
SSC_{13}	RK edge to RF (RK output)	–	-2.1(1)	1.9(1)	ns

Note: 1. For output signals (TF, TD, RF), min and max access times are defined. The min access time is the time between the TK (or RK) edge and the signal change. The max access timing is the time between the TK edge and the signal stabilization. Figure 56-41 illustrates min and max accesses for SSC0. The same applies for SSC1, SSC4, and SSC7, SSC10 and SSC13.

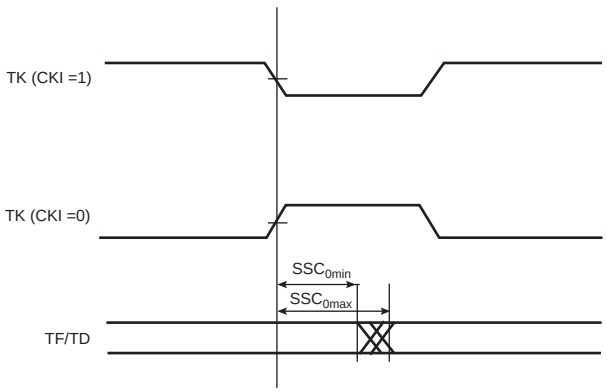
Downloaded from Arrow.com.

Table 56-72. SSC Timings with 1.8V Peripheral Supply

Symbol	Parameter	Condition	Min	Max	Unit
Transmitter					
SSC_0	TK edge to TF/TD (TK output, TF output)	–	-5.7(1)	5.3(1)	ns
SSC_1	TK edge to TF/TD (TK input, TF output)	–	3.6(1)	16.8(1)	ns
SSC_2	TF setup time before TK edge (TK output)	–	17.3	–	ns
SSC_3	TF hold time after TK edge (TK output)	–	0	–	ns
SSC_4	TK edge to TF/TD (TK output, TF input)	–	-5.7(1)	3.1(1)	ns
		STTDLY = 0 START = 4, 5 or 7	$-5.7 + (2 \times t_{CPMCK})$(1)	$3.1 + (2 \times t_{CPMCK})$(1)	
SSC_5	TF setup time before TK edge (TK input)	–	0	–	ns
SSC_6	TF hold time after TK edge (TK input)	–	t_{CPMCK}	–	ns
SSC_7	TK edge to TF/TD (TK input, TF input)	–	3.6(1)	14.7(1)	ns
		STTDLY = 0 START = 4, 5 or 7	$3.6 + (3 \times t_{CPMCK})$(1)	$14.7 + (3 \times t_{CPMCK})$(1)	
Receiver					
SSC_8	RF/RD setup time before RK edge (RK input)	–	0	–	ns
SSC_9	RF/RD hold time after RK edge (RK input)	–	t_{CPMCK}	–	ns
SSC_{10}	RK edge to RF (RK input)	–	3.5(1)	12.1(1)	ns
SSC_{11}	RF/RD setup time before RK edge (RK output)	–	$13.5 - t_{CPMCK}$	–	ns
SSC_{12}	RF/RD hold time after RK edge (RK output)	–	$t_{CPMCK} - 2.9$	–	ns
SSC_{13}	RK edge to RF (RK output)	–	-2.8(1)	2.6(1)	ns

Note: 1. For output signals (TF, TD, RF), min and max access times are defined. The min access time is the time between the TK (or RK) edge and the signal change. The max access timing is the time between the TK edge and the signal stabilization. Figure 56-41 illustrates min and max accesses for SSC0. The same applies for SSC1, SSC4, and SSC7, SSC10 and SSC13.

Figure 56-41. Min and Max Access Time of Output Signals



Downloaded from Arrow.com.

56.13.1.13 ISI Timings

Timing Conditions

Timings are given assuming the load capacitance in Table 56-73.

Table 56-73. Load Capacitance

Supply	C_L Max
3.3V	30 pF
1.8V	20 pF

Timing Extraction

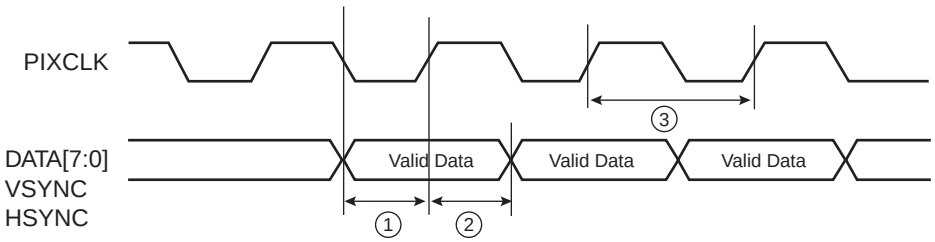
Table 56-74. ISI Timings with Peripheral Supply 3.3V

Symbol	Parameter	Min	Max	Unit
ISI_1	DATA/VSYNC/HSYNC setup time	1.5	–	ns
ISI_2	DATA/VSYNC/HSYNC hold time	-1.2	–	ns
ISI_3	PIXCLK frequency	–	75	MHz

Table 56-75. ISI Timings with Peripheral Supply 1.8V

Symbol	Parameter	Min	Max	Unit
ISI_1	DATA/VSYNC/HSYNC setup time	1.8	–	ns
ISI_2	DATA/VSYNC/HSYNC hold time	-1.4	–	ns
ISI_3	PIXCLK frequency	–	75	MHz

Figure 56-42. ISI Timing Diagram



Downloaded from Arrow.com.

56.14 Timings for STH Conditions

The timings are applicable under the conditions inTable 56-76.

Table 56-76. STH Timings Conditions

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{DDCORE}	DC Supply Core	–	1.2	–	1.32	V

56.14.1 AC Characteristics

56.14.1.1 Processor Clock Characteristics

Table 56-77. Processor Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Unit
$1/(t_{CPPCK})$	Processor Clock Frequency	–	–	300	MHz

56.14.1.2 Master Clock Characteristics

Table 56-78. Master Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Unit
$1/(t_{CPMCK})$	Master Clock Frequency	–	–	150	MHz

Downloaded from Arrow.com.

56.14.1.3 I/O Characteristics

Criteria used to define the maximum frequency of the I/Os:

- Output duty cycle (40%-60%)
- Minimum output swing: 100 mV to VDDIO - 100 mV
- Addition of rising and falling time inferior to 75% of the period.

Table 56-79. I/O Characteristics

Symbol	Parameter	Conditions			Min	Max	Unit
		Load	V_{DDIO}	Drive Level			
FreqMax1	Pin Group 1 (1) Maximum output frequency	10 pF	1.7V	Low	–	40	MHz
				High	–	65	
		25 pF		Low	–	20	
				High	–	33	
		10 pF	2.7V	Low	–	65	
				High	–	115	
		25 pF		Low	–	28	
				High	–	55	
$PulseminH_1$	Pin Group 1 (1) High Level Pulse Width	10 pF	1.7V	High	6.1	9.2	ns
$PulseminL_1$	Pin Group 1 (1) Low Level Pulse Width	10 pF	1.7V	High	6.1	9.2	ns
FreqMax2	Pin Group 2 (2)Maximum output frequency	10 pF	3.0V	High	–	150	MHz
$PulseminH_2$	Pin Group 2 (2) High Level Pulse Width	10 pF	3.0V	High	3.0	3.6	ns
$PulseminL_2$	Pin Group 2 (2) Low Level Pulse Width	10 pF	3.0V	High	3.0	3.6	ns
FreqMax3	Pin Group3(3) Maximum output frequency	30 pF	3.0V	High	–	85	MHz
$PulseminH_3$	Pin Group 3 (3) High Level Pulse Width	30 pF	3.0V	High	5.3	6.5	ns
$PulseminL_3$	Pin Group 3 (3) Low Level Pulse Width	30 pF	3.0V	High	5.3	6.5	ns
FreqMax4	Pin Group 4(4)Maximum output frequency	40 pF	2.7V	–	–	58	MHz
$PulseminH_4$	Pin Group 4(4) High Level Pulse Width	40 pF	2.7V	–	6.9	9.9	ns
$PulseminL_4$	Pin Group 4(4) Low Level Pulse Width	40 pF	2.7V	–	6.9	9.9	ns

Notes:
1. Pin Group 1 = GPIO, CLOCK
2. Pin Group 2 = GPIO_CLK
3. Pin Group 3 = GPIO_AD
4. Pin Group 4 = GPIO_MLB

Downloaded from Arrow.com.

56.14.1.4 QSPI Characteristics

Figure 56-43. QSPI Master Mode with (CPOL= NCPHA = 0) or (CPOL= NCPHA= 1)

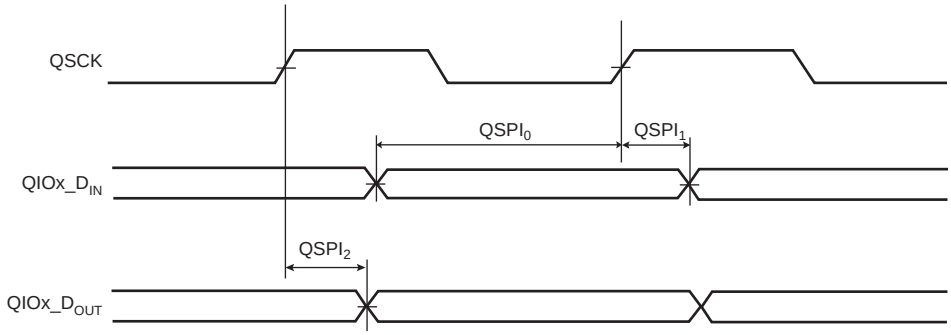
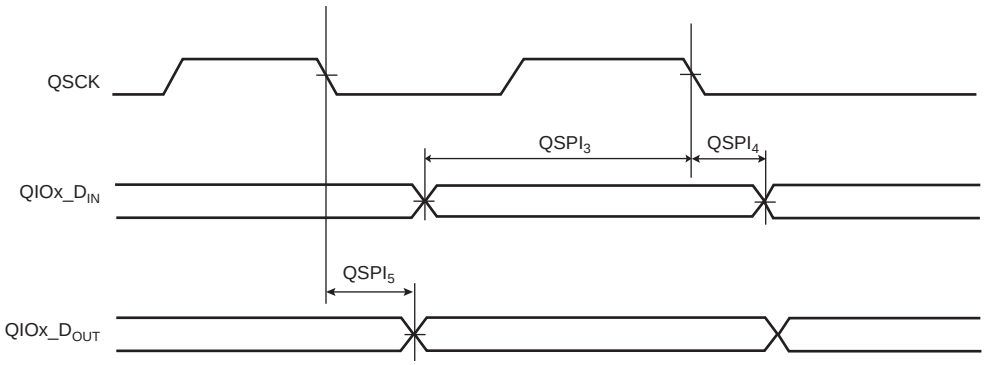


Figure 56-44. QSPI Master Mode with (CPOL = 0 and NCPHA=1) or (CPOL=1 and NCPHA= 0)



Downloaded from Arrow.com.

Maximum QSPI Frequency

The following formulas give maximum QSPI frequency in Master read and write modes.

Master Write Mode

The QSPI sends data to a slave device only, e.g. an LCD. The limit is given by $QSPI_2$ (or $QSPI_5$) timing. Since it gives a maximum frequency above the maximum pad speed (see Section 56.13.1.3 “I/O Characteristics”), the max QSPI frequency is the one from the pad.

Master Read Mode

$$f_{SPCK}max = \frac{1}{QSPI_0(or\ QSPI_3) + t_{valid}}$$

t_{valid} is the slave time response to output data after detecting a QSCK edge.

For a non-volatile memory with t_{valid} (or t_v) = 12 ns, f_{SCK}max = 69 MHz at V_{DDIO} = 3.3V.

QSPI Timings

Timings are given in the following domains:

- 1.8V domain: VDDIO from 1.7V to 1.95V, maximum external capacitor = 20 pF
- 3.3V domain: VDDIO from 2.85V to 3.6V, maximum external capacitor = 40 pF

Table 56-80. QSPI Timings

Symbol	Parameter	Conditions	Min	Max	Unit
$QSPI_0$	QIOx data in to QSCK rising edge (input setup time)	3.3V domain	2.1	–	ns
		1.8V domain	2.9	–	ns
$QSPI_1$	QIOx data in to QSCK rising edge (input hold time)	3.3V domain	0	–	ns
		1.8V domain	0	–	ns
$QSPI_2$	QSCK rising edge to QIOx data out valid	3.3V domain	-1.3	1.7	ns
		1.8V domain	-2.5	2.9	ns
$QSPI_3$	QIOx data in to QSCK falling edge (input setup time)	3.3V domain	2.5	–	ns
		1.8V domain	2.9	–	ns
$QSPI_4$	QIOx data in to QSCK falling edge(input hold time)	3.3V domain	0	–	ns
		1.8V domain	0	–	ns
$QSPI_5$	QSCK falling edge to QIOx data out valid	3.3V domain	-1.5	1.7	ns
		1.8V domain	-2.4	3.0	ns

Downloaded from Arrow.com.

56.14.1.5 SPI Characteristics

In Figure 56-46, "SPI Master Mode with (CPOL= NCPHA = 0) or (CPOL= NCPHA= 1)" and Figure 56-46, "SPI Master Mode with (CPOL= NCPHA = 0) or (CPOL= NCPHA= 1)" below, the MOSI line shifting edge is represented with a hold time equal to 0. However, it is important to note that for this device, the MISO line is sampled prior to the MOSI line shifting edge. As shown in Figure 56-45, "MISO Capture in Master Mode", the device sampling point extends the propagation delay (t_p) for slave and routing delays to more than half the SPI clock period, whereas the common sampling point allows only less than half the SPI clock period.

As an example, an SPI Slave working in Mode 0 can be safely driven if the SPI Master is configured in Mode 0.

Figure 56-45. MISO Capture in Master Mode

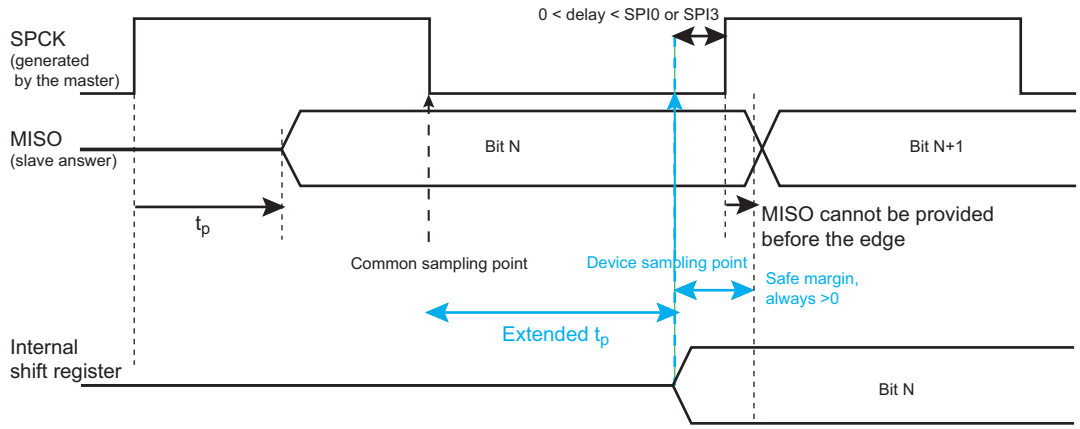
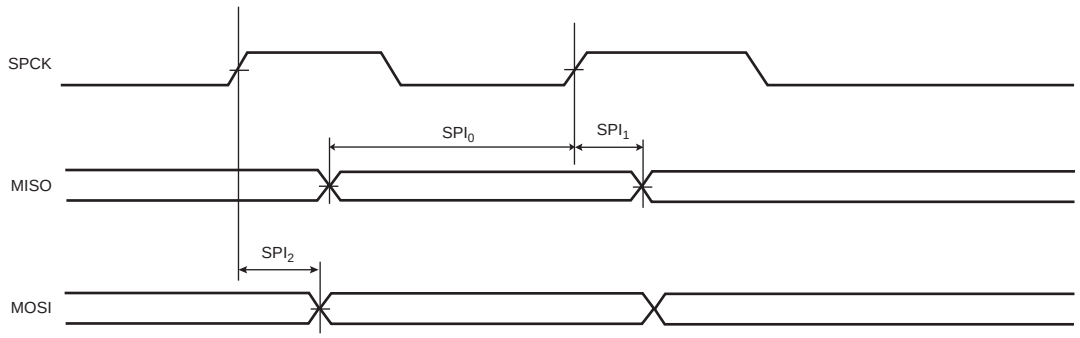


Figure 56-46. SPI Master Mode with (CPOL= NCPHA = 0) or (CPOL= NCPHA= 1)



Downloaded from Arrow.com.

Figure 56-47. SPI Master Mode with (CPOL = 0 and NCPHA=1) or (CPOL=1 and NCPHA= 0)

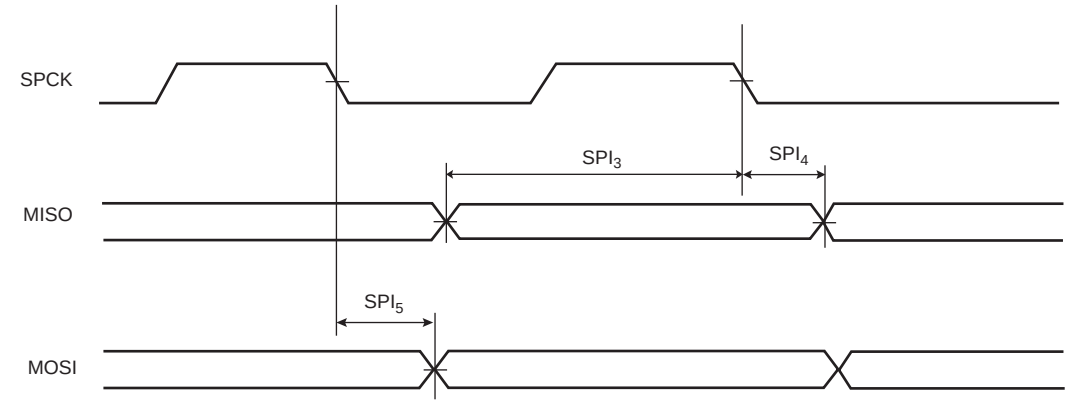


Figure 56-48. SPI Slave Mode with (CPOL=0 and NCPHA=1) or (CPOL=1 and NCPHA=0)

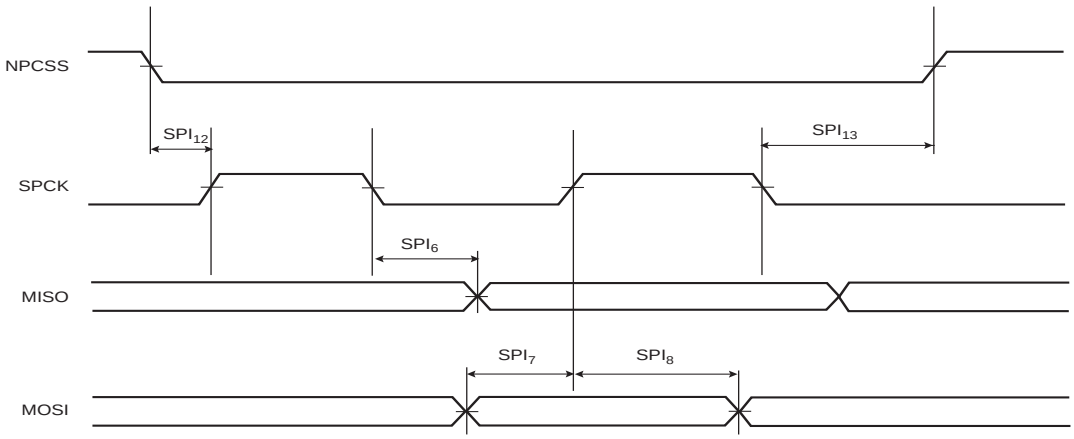
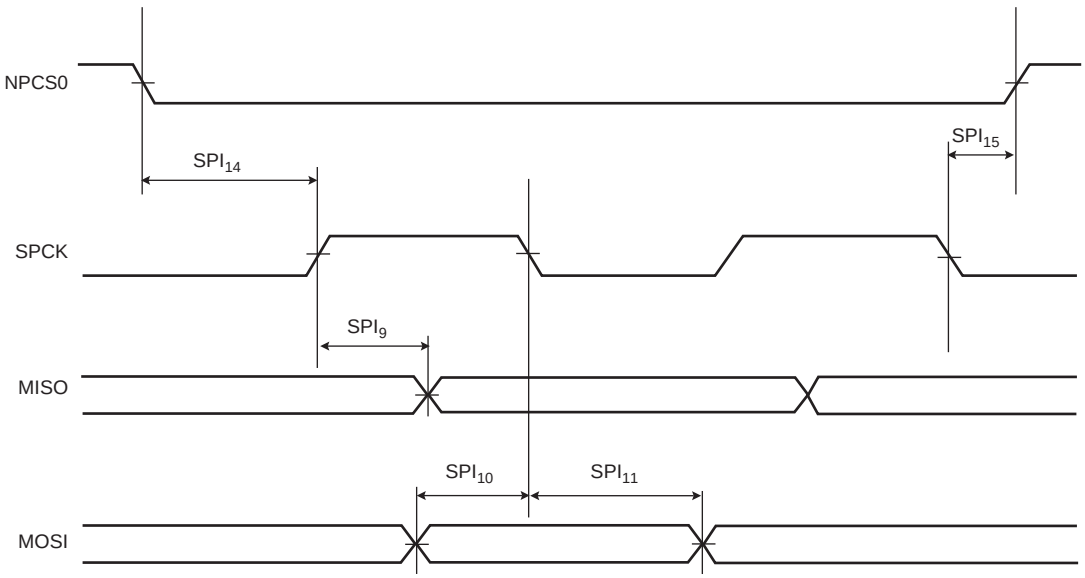


Figure 56-49. SPI Slave Mode with (CPOL = NCPHA = 0) or (CPOL= NCPHA= 1)



Downloaded from Arrow.com.

Maximum SPI Frequency

The following formulas give maximum SPI frequency in Master read and write modes and in Slave read and write modes.

Master Write Mode

The SPI sends data to a slave device only, e.g. an LCD. The limit is given by SPI_2 (or SPI_5) timing. Since it gives a maximum frequency above the maximum pad speed (see Section 56.13.1.3 “I/O Characteristics”), the max SPI frequency is the one from the pad.

Master Read Mode

$$f_{SPCK}max = \frac{1}{SPI_0(\text{or } SPI_3) + t_{valid}}$$

t_{valid} is the slave time response to output data after detecting an SPCK edge.

For a non-volatile memory with t_{valid} (or t_v) = 5 ns, $f_{SPCK}max$ = 63 MHz at V_{DDIO} = 3.3V.

Slave Read Mode

In slave mode, SPCK is the input clock for the SPI. The max SPCK frequency is given by setup and hold timings SPI_7/SPI_8(or SPI_{10}/SPI_{11}). Since this gives a frequency well above the pad limit, the limit in slave read mode is given by SPCK pad.

Slave Write Mode

$$f_{SPCK}max = \frac{1}{2x(SPI_{6max}(\text{or } SPI_{9max}) + t_{setup})}$$

t_{setup} is the setup time from the master before sampling data.

Downloaded from Arrow.com.

SPI Timings

Timings are given in the following domains:

- 1.8V domain: VDDIO from 1.7V to 1.95V, maximum external capacitor = 20 pF
- 3.3V domain: VDDIO from 2.85V to 3.6V, maximum external capacitor = 40 pF.

Table 56-81. SPI Timings

Symbol	Parameter	Conditions	Min	Max	Unit
SPI_0	MISO Setup time before SPCK rises (master)	3.3V domain	10.8	–	ns
		1.8V domain	12.6	–	ns
SPI_1	MISO Hold time after SPCK rises (master)	3.3V domain	0	–	ns
		1.8V domain	0	–	ns
SPI_2	SPCK rising to MOSI Delay (master)	3.3V domain	-3.4	2.1	ns
		1.8V domain	-3.6	2.6	ns
SPI_3	MISO Setup time before SPCK falls (master)	3.3V domain	11.0	–	ns
		1.8V domain	13.2	–	ns
SPI_4	MISO Hold time after SPCK falls (master)	3.3V domain	0	–	ns
		1.8V domain	0	–	ns
SPI_5	SPCK falling to MOSI Delay (master)	3.3V domain	-3.2	2.0	ns
		1.8V domain	-3.0	2.8	ns
SPI_6	SPCK falling to MISO Delay (slave)	3.3V domain	3.0	10.6	ns
		1.8V domain	3.5	12.9	ns
SPI_7	MOSI Setup time before SPCK rises (slave)	3.3V domain	0.9	–	ns
		1.8V domain	1.6	–	ns
SPI_8	MOSI Hold time after SPCK rises (slave)	3.3V domain	0.6	–	ns
		1.8V domain	1.2	–	ns
SPI_9	SPCK rising to MISO Delay (slave)	3.3V domain	3.0	10.6	ns
		1.8V domain	3.4	12.5	ns
SPI_{10}	MOSI Setup time before SPCK falls (slave)	3.3V domain	0.9	–	ns
		1.8V domain	1.6	–	ns
SPI_{11}	MOSI Hold time after SPCK falls (slave)	3.3V domain	0.6	–	ns
		1.8V domain	1.2	–	ns
SPI_{12}	NPCS setup to SPCK rising (slave)	3.3V domain	3.2	–	ns
		1.8V domain	3.4	–	ns
SPI_{13}	NPCS hold after SPCK falling (slave)	3.3V domain	0	–	ns
		1.8V domain	0	–	ns
SPI_{14}	NPCS setup to SPCK falling (slave)	3.3V domain	3.2	–	ns
		1.8V domain	3.0	–	ns
SPI_{15}	NPCS hold after SPCK falling (slave)	3.3V domain	0	–	ns
		1.8V domain	0	–	ns

Note that in SPI Master mode, data (MISO) is not sampled on the opposite edge where the data clocks out (MOSI), but the same edge is used. See Figure 56-20 and Figure 56-21.

Downloaded from Arrow.com.

56.14.1.6 HSMCI Timings

The High Speed MultiMedia Card Interface (HSMCI) supports the MultiMedia Card (e.MMC) Specification V4.3, the SD Memory Card Specification V2.0, the SDIO V2.0 specification and CE-ATA V1.1.

56.14.1.7 SDRAM Timings

To achieve 150 MHz on the SDRAM interface, the user must use SDRAM devices that satisfy the timings of standard SDR-200 and LP-SDR-200 modules. SDR-200 and LPSDR-200 timings are specified by the JEDEC standard.

Downloaded from Arrow.com.

56.14.1.8 SMC Timings

Timings are given in the following domains:

- 1.8V domain: VDDIO from 1.7V to 1.95V, maximum external capacitor = 30 pF
- 3.3V domain: VDDIO from 2.85V to 3.6V, maximum external capacitor = 50 pF

Timings are given assuming a load capacitance on data, control and address pads:

In the tables that follow, t_{CPMCK} is MCK period.

Read Timings

Table 56-82. SMC Read Signals - NRD Controlled (READ_MODE = 1)

Symbol	VDDIO Supply / Parameter	1.8V Domain (Min)	3.3V Domain (Min)	1.8V Domain (Max)	3.3V Domain (Max)	Unit
	NO HOLD Settings (NRD_HOLD = 0)					
SMC_1	Data Setup before NRD High	15.7	12.5	—	—	ns
SMC_2	Data Hold after NRD High	0	0	—	—	ns
	HOLD Settings (NRD_HOLD ≠ 0)					
SMC_3	Data Setup before NRD High	14.0	10.7	—	—	ns
SMC_4	Data Hold after NRD High	0	0	—	—	ns
	HOLD or NO HOLD Settings (NRD_HOLD ≠ 0, NRD_HOLD = 0)					
SMC_5	A0–A22 Valid before NRD High	(NRD_SETUP + NRD_PULSE) × t_{CPMCK} - 4.9	(NRD_SETUP + NRD_PULSE) × t_{CPMCK} - 4.1	—	—	ns
SMC_6	NCS low before NRD High	(NRD_SETUP + NRD_PULSE - NCS_RD_SETUP) × t_{CPMCK} - 3.5	(NRD_SETUP + NRD_PULSE - NCS_RD_SETUP) × t_{CPMCK} - 2.6	—	—	ns
SMC_7	NRD Pulse Width	NRD_PULSE × t_{CPMCK} - 0.8	NRD_PULSE × t_{CPMCK} - 0.3	—	—	ns

Table 56-83. SMC Read Signals - NCS Controlled (READ_MODE = 0)

Symbol	VDDIO Supply / Parameter	1.8V Domain (Min)	3.3V Domain (Min)	1.8V Domain (Max)	3.3V Domain (Max)	Unit
	NO HOLD Settings (NCS_RD_HOLD = 0)					
SMC_8	Data Setup before NCS High	24.5	19.5	—	—	ns
SMC_9	Data Hold after NCS High	0	0	—	—	ns
	HOLD Settings (NCS_RD_HOLD ≠ 0)					
SMC_{10}	Data Setup before NCS High	15.3	10.1	—	—	ns
SMC_{11}	Data Hold after NCS High	0	0	—	—	ns
	HOLD or NO HOLD Settings (NCS_RD_HOLD ≠ 0, NCS_RD_HOLD = 0)					
SMC_{12}	A0–A22 valid before NCS High	(NCS_RD_SETUP + NCS_RD_PULSE) × t_{CPMCK} - 6.8	(NCS_RD_SETUP + NCS_RD_PULSE) × t_{CPMCK} - 3.8	—	—	ns
SMC_{13}	NRD low before NCS High	(NCS_RD_SETUP + NCS_RD_PULSE - NRD_SETUP) × t_{CPMCK} - 7.1	(NCS_RD_SETUP + NCS_RD_PULSE - NRD_SETUP) × t_{CPMCK} - 4.1	—	—	ns
SMC_{14}	NCS Pulse Width	NCS_RD_PULSE length × t_{CPMCK} - 1.1	NCS_RD_PULSE length × t_{CPMCK} - 0.3	—	—	ns

Downloaded from Arrow.com.

Write Timings

Table 56-84. SMC Write Signals - NWE Controlled (WRITE_MODE = 1)

Symbol	Parameter	VDDIO Supply 1.8V Domain Min	3.3V Domain Min	1.8V Domain Max	3.3V Domain Max	Unit
HOLD or NO HOLD Settings (NWE_HOLD ≠ 0, NWE_HOLD = 0)						
SMC_{15}	Data Out Valid before NWE High	NWE_PULSE × t_{CPMCK} - 5.0	NWE_PULSE × t_{CPMCK} - 4.2	—	—	ns
SMC_{16}	NWE Pulse Width	NWE_PULSE × t_{CPMCK} - 0.8	NWE_PULSE × t_{CPMCK} - 0.4	—	—	ns
SMC_{17}	A0–A22 valid before NWE low	NWE_SETUP × t_{CPMCK} - 4.6	NWE_SETUP × t_{CPMCK} - 3.9	—	—	ns
SMC_{18}	NCS low before NWE high	(NWE_SETUP - NCS_RD_SETUP + NWE_PULSE) × t_{CPMCK} - 3.3	(NWE_SETUP - NCS_RD_SETUP + NWE_PULSE) × t_{CPMCK} - 2.4	—	—	ns
HOLD Settings (NWE_HOLD ≠ 0)						
SMC_{19}	NWE High to Data OUT, NBS0/A0 NBS1, A1, A2–A25 change	NWE_HOLD × t_{CPMCK} - 4.4	NWE_HOLD × t_{CPMCK} - 3.8	—	—	ns
SMC_{20}	NWE High to NCS Inactive (1)	(NWE_HOLD - NCS_WR_HOLD) × t_{CPMCK} - 3.8	(NWE_HOLD - NCS_WR_HOLD) × t_{CPMCK} - 3.5	—	—	ns
NO HOLD Settings (NWE_HOLD = 0)						
SMC_{21}	NWE High to Data OUT, NBS0/A0 NBS1, A1, A2–A25, NCS change(1)	2.1	1.5	—	—	ns

Notes: 1. Hold length = total cycle duration - setup duration - pulse duration. “hold length” is for “NCS_WR_HOLD length” or “NWE_HOLD length”.

Table 56-85. SMC Write NCS Controlled (WRITE_MODE = 0)

Symbol	Parameter	VDDIO Supply 1.8V Domain Min	3.3V Domain Min	1.8V Domain Max	3.3V Domain Max	Unit
SMC_{22}	Data Out Valid before NCS High	NCS_WR_PULSE × t_{CPMCK} - 6.8	NCS_WR_PULSE × t_{CPMCK} - 3.8	—	—	ns
SMC_{23}	NCS Pulse Width	NCS_WR_PULSE × t_{CPMCK} - 1.1	NCS_WR_PULSE × t_{CPMCK} - 0.3	—	—	ns
SMC_{24}	A0–A22 valid before NCS low	NCS_WR_SETUP × t_{CPMCK} - 7.2	NCS_WR_SETUP × t_{CPMCK} - 4.4	—	—	ns
SMC_{25}	NWE low before NCS high	(NCS_WR_SETUP - NWE_SETUP + NCS pulse) × t_{CPMCK} - 7.2	(NCS_WR_SETUP - NWE_SETUP + NCS pulse) × t_{CPMCK} - 4.4	—	—	ns
SMC_{26}	NCS High to Data Out, A0–A25, change	NCS_WR_HOLD × t_{CPMCK} - 5.1	NCS_WR_HOLD × t_{CPMCK} - 3.1	—	—	ns
SMC_{27}	NCS High to NWE Inactive	(NCS_WR_HOLD - NWE_HOLD) × t_{CPMCK} - 3.7	(NCS_WR_HOLD - NWE_HOLD) × t_{CPMCK} - 2.2	—	—	ns

Downloaded from Arrow.com.

Figure 56-50. SMC Timings - NCS Controlled Read and Write

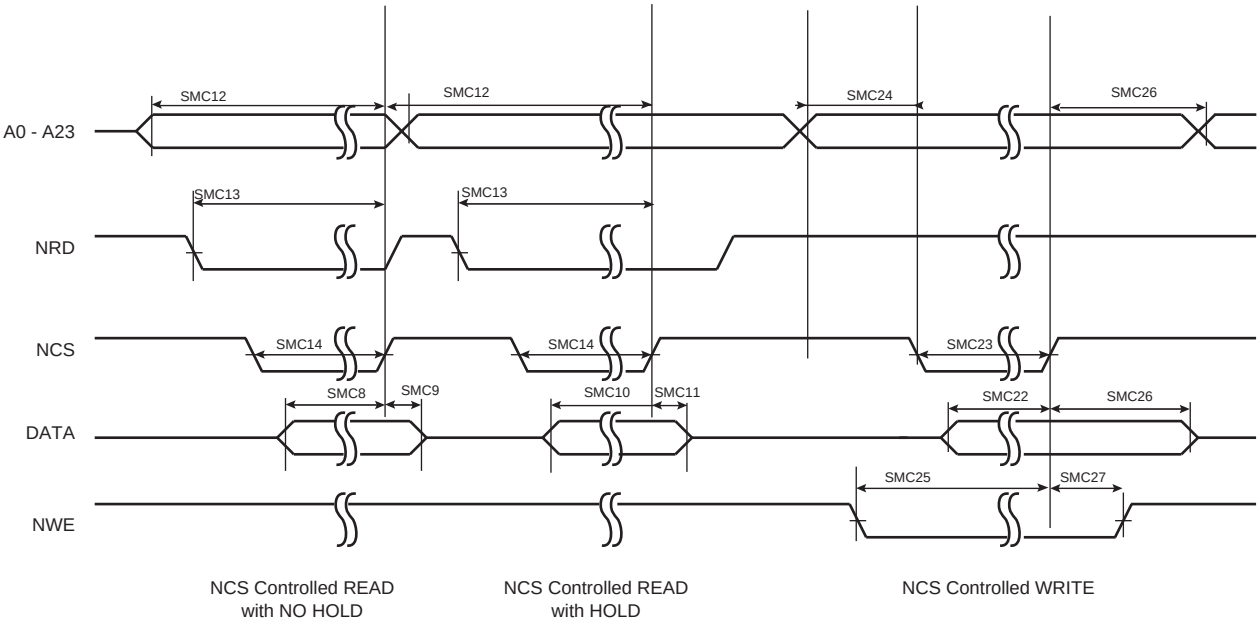
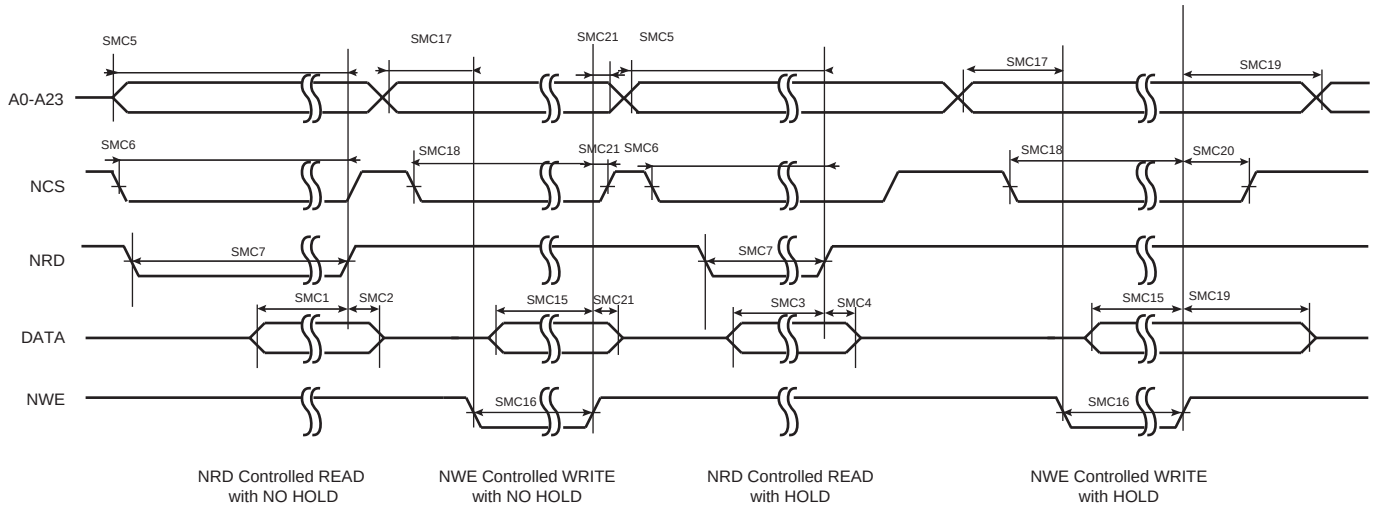


Figure 56-51. SMC Timings - NRD Controlled Read and NWE Controlled Write



Downloaded from Arrow.com.

56.14.1.9 USART in SPI Mode Timings

Figure 56-52. USART SPI Master Mode

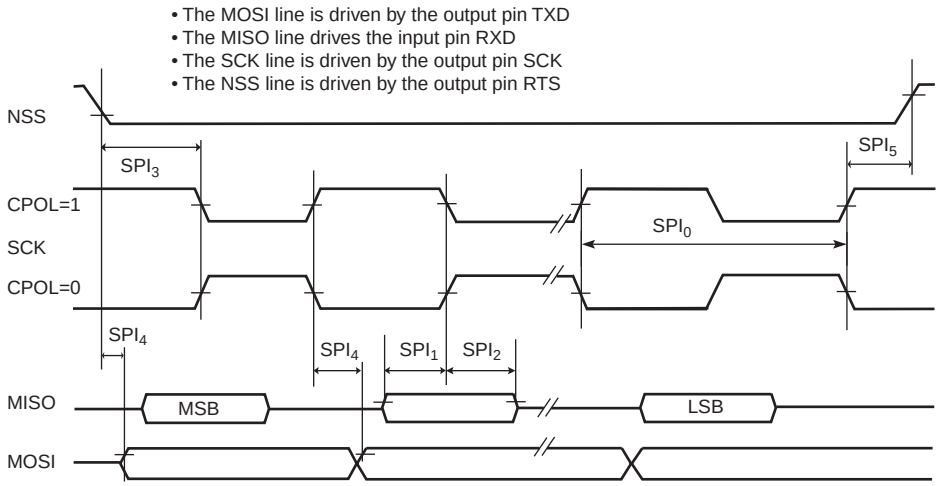
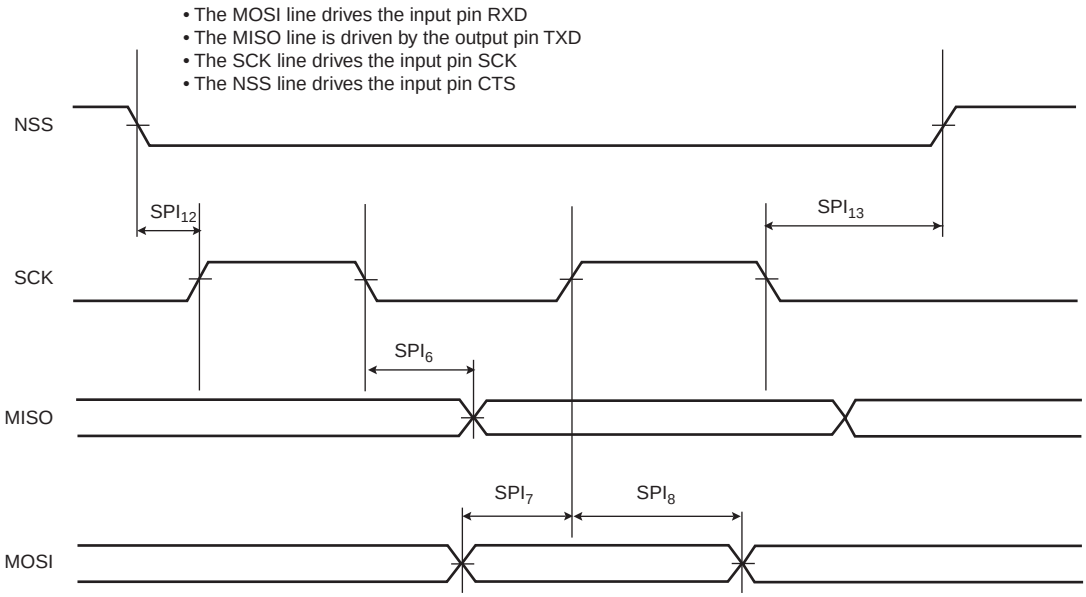
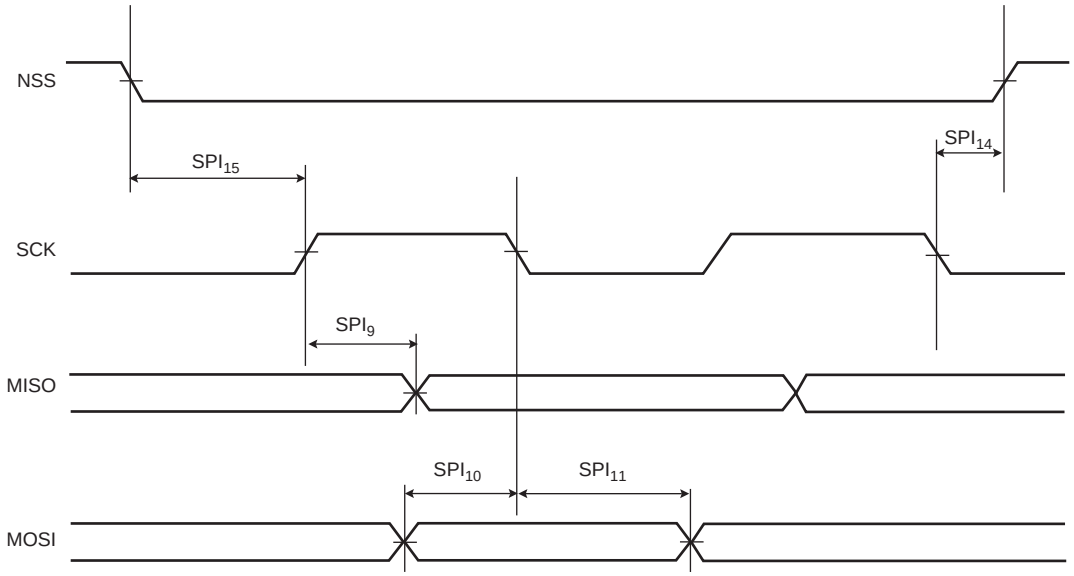


Figure 56-53. USART SPI Slave Mode (Mode 1 or 2)



Downloaded from Arrow.com.

Figure 56-54. USART SPI Slave Mode (Mode 0 or 3)



Downloaded from Arrow.com.

USART SPI TImings

Timings are given in the following domains:

- 1.8V domain: VDDIO from 1.7V to 1.95V, maximum external capacitor = 20 pF
- 3.3V domain: VDDIO from 2.85V to 3.6V, maximum external capacitor = 40 pF

Table 56-86. USART SPI Timings

Symbol	Parameter	Conditions	Min	Max	Unit
Master Mode					
SPI_0	SCK Period	1.8V domain 3.3V domain	MCK/6	–	ns
SPI_1	Input Data Setup Time	1.8V domain 3.3V domain	2.3 2.1	–	ns
SPI_2	Input Data Hold Time	1.8V domain 3.3V domain	0.5 0.2	–	ns
SPI_3	Chip Select Active to Serial Clock	1.8V domain 3.3V domain	-1.1 -0.9	–	ns
SPI_4	Output Data Setup Time	1.8V domain 3.3V domain	-2.1 -2.0	10.5 9.9	ns
SPI_5	Serial Clock to Chip Select Inactive	1.8V domain 3.3V domain	-2.6 -2.5	-2.1 -2.0	ns
Slave Mode					
SPI_6	SCK falling to MISO	1.8V domain 3.3V domain	3.5 2.9	15.6 12.3	ns
SPI_7	MOSI Setup time before SCK rises	1.8V domain 3.3V domain	1.9 1.8	–	ns
SPI_8	MOSI Hold time after SCK rises	1.8V domain 3.3V domain	0.5 0.2	–	ns
SPI_9	SCK rising to MISO	1.8V domain 3.3V domain	3.5 3.0	14.9 11.9	ns
SPI_{10}	MOSI Setup time before SCK falls	1.8V domain 3.3V domain	1.6 1.8	–	ns
SPI_{11}	MOSI Hold time after SCK falls	1.8V domain 3.3V domain	0.9 0.3	–	ns
SPI_{12}	NPCS0 setup to SCK rising	1.8V domain 3.3V domain	1.7 0.9	–	ns
SPI_{13}	NPCS0 hold after SCK falling	1.8V domain 3.3V domain	1.3 0.5	–	ns
SPI_{14}	NPCS0 setup to SCK falling	1.8V domain 3.3V domain	1.3 0.9	–	ns
SPI_{15}	NPCS0 hold after SCK rising	1.8V domain 3.3V domain	0.9 1.0	–	ns

Downloaded from Arrow.com.

56.14.1.10 Two-wire Serial Interface Characteristics

Table 56-87 describes the requirements for devices connected to the Two-wire Serial Bus.

For timing symbols, refer to Figure 56-29.

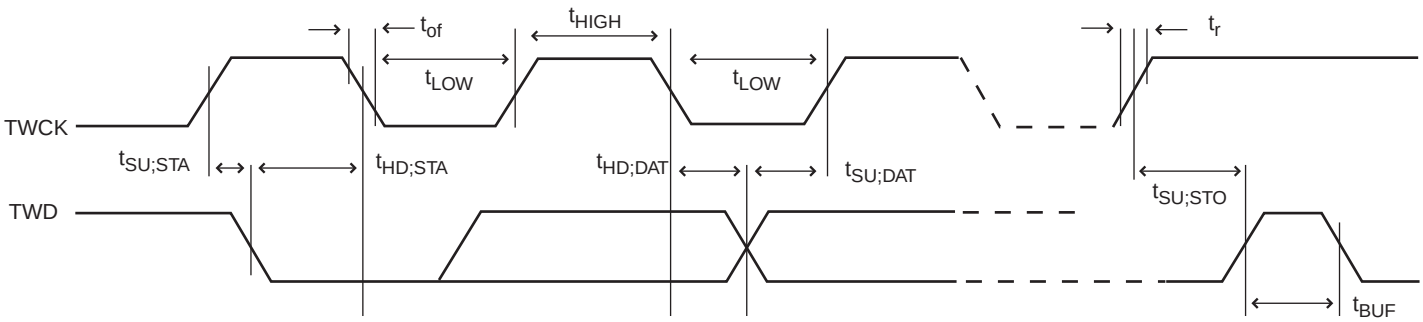
Table 56-87. Two-wire Serial Bus Requirements

Symbol	Parameter	Condition	Min	Max	Unit
V_{IL}	Low-level Input Voltage	–	-0.3	0.3 V_{DDIO}	V
V_{IH}	High-level Input Voltage	–	0.7 × V_{DDIO}	V_{CC} + 0.3	V
V_{hys}	Hysteresis of Schmitt Trigger Inputs	–	0.150	–	V
V_{OL}	Low-level Output Voltage	3 mA sink current	–	0.4	V
t_R	Rise Time for both TWD and TWCK	–	20 + 0.1C_b (1)(2)	300	ns
t_{OF}	Output Fall Time from V_{IHmin} to V_{ILmax}	10 pF < C_b < 400 pF Figure 56-29	20 + 0.1C_b (1)(2)	250	ns
C_i (1)	Capacitance for each I/O Pin	–	–	10	pF
f_{TWCK}	TWCK Clock Frequency	–	0	400	kHz
R_P	Value of Pull-up resistor	f_{TWCK} ≤ 100 kHz	(V_{DDIO} - 0.4V) ÷ 3mA	1000ns ÷ C_b	Ω
		f_{TWCK} > 100 kHz		300ns ÷ C_b	Ω
t_{LOW}	Low Period of the TWCK clock	f_{TWCK} ≤ 100 kHz	(3)	–	µs
		f_{TWCK} > 100 kHz	(3)	–	µs
t_{HIGH}	High period of the TWCK clock	f_{TWCK} ≤ 100 kHz	(4)	–	µs
		f_{TWCK} > 100 kHz	(4)	–	µs
$t_{HD;STA}$	Hold Time (repeated) START Condition	f_{TWCK} ≤ 100 kHz	t_{HIGH}	–	µs
		f_{TWCK} > 100 kHz	t_{HIGH}	–	µs
$t_{SU;STA}$	Set-up time for a repeated START condition	f_{TWCK} ≤ 100 kHz	t_{HIGH}	–	µs
		f_{TWCK} > 100 kHz	t_{HIGH}	–	µs
$t_{HD;DAT}$	Data hold time	f_{TWCK} ≤ 100 kHz	0	3 × t_{CPMCK} (5)	µs
		f_{TWCK} > 100 kHz	0	3 × t_{CPMCK} (5)	µs
$t_{SU;DAT}$	Data setup time	f_{TWCK} ≤ 100 kHz	t_{LOW} - 3 × t_{CPMCK} (5)	–	ns
		f_{TWCK} > 100 kHz	t_{LOW} - 3 × t_{CPMCK} (5)	–	ns
$t_{SU;STO}$	Setup time for STOP condition	f_{TWCK} ≤ 100 kHz	t_{HIGH}	–	µs
		f_{TWCK} > 100 kHz	t_{HIGH}	–	µs
$t_{HD;STA}$	Hold Time (repeated) START Condition	f_{TWCK} ≤ 100 kHz	t_{HIGH}	–	µs
		f_{TWCK} > 100 kHz	t_{HIGH}	–	µs

Notes:
1. Required only for f_{TWCK} > 100 kHz.
2. C_b = capacitance of one bus line in pF. Per I^2C standard, $C_b$max = 400pF
3. The TWCK low period is defined as follows: t_{LOW} = ((CLDIV × 2^{CKDIV}) + 4) × t_{MCK}
4. The TWCK high period is defined as follows: t_{HIGH} = ((CHDIV × 2^{CKDIV}) + 4) × t_{MCK}
5. t_{CPMCK} = MCK bus period.

Downloaded from Arrow.com.

Figure 56-55. Two-wire Serial Bus Timing



Downloaded from Arrow.com.

56.14.1.11 GMAC Characteristics

Timing Conditions

Table 56-88. Load Capacitance on Data, Clock Pads

Supply	C_L	
	Max	Min
3.3V	20 pF	0 pF

Timing Constraints

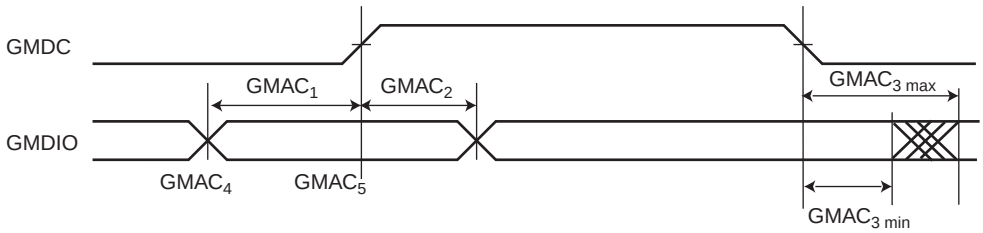
The GMAC must be constrained so as to satisfy the timings of standard given in Table 56-89 and Table 56-90, in MAX and STH corners**.**

Table 56-89. GMAC Signals Relative to GMDC

Symbol	Parameter	Min	Max	Unit
$GMAC_1$	Setup for GMDIO from GMDC rising	10	–	ns
$GMAC_2$	Hold for GMDIO from GMDC rising	10	–	
$GMAC_3$	GMDIO toggling from GMDC falling	0(1)	10(1)	

Note: 1. For GMAC output signals, min and max access time are defined. The min access time is the time between the GMDC falling edge and the signal change. The max access timing is the time between the GMDC falling edge and the signal stabilizes. Figure 56-56 illustrates min and max accesses for $GMAC_3$.

Figure 56-56. Min and Max Access Time of GMAC Output Signals



Downloaded from Arrow.com.

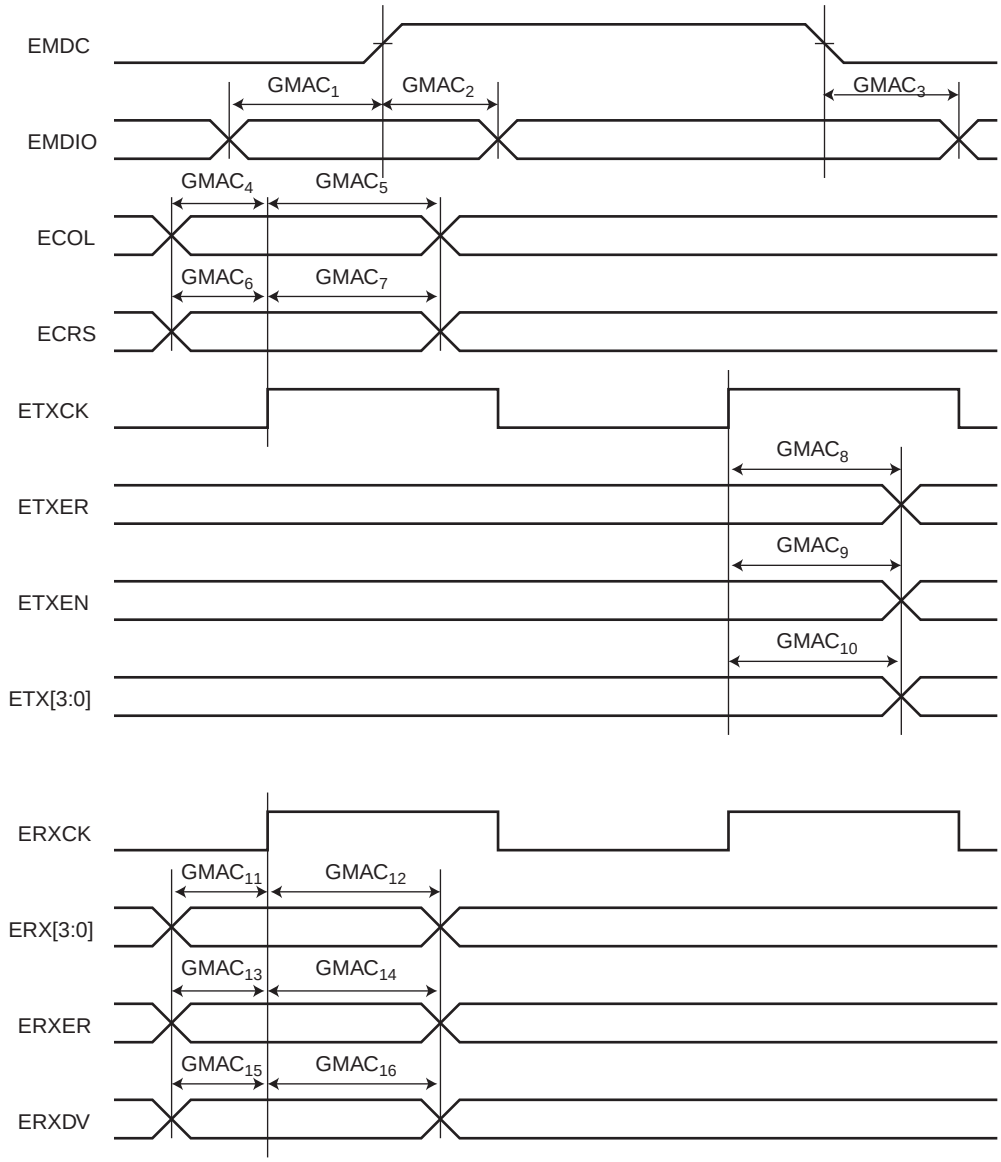
MII Mode

Table 56-90. GMAC MII Mode Timings

Symbol	Parameter	Min	Max	Unit
$GMAC_4$	Setup for GCOL from GTXCK rising	10	–	ns
$GMAC_5$	Hold for GCOL from GTXCK rising	10	–	
$GMAC_6$	Setup for GCRS from GTXCK rising	10	–	
$GMAC_7$	Hold for GCRS from GTXCK rising	10	–	
$GMAC_8$	GTXER toggling from GTXCK rising	10	25	
$GMAC_9$	GTXEN toggling from GTXCK rising	10	25	
$GMAC_{10}$	GTX toggling from GTXCK rising	10	25	
$GMAC_{11}$	Setup for GRX from GRXCK	10	–	
$GMAC_{12}$	Hold for GRX from GRXCK	10	–	
$GMAC_{13}$	Setup for GRXER from GRXCK	10	–	
$GMAC_{14}$	Hold for GRXER from GRXCK	10	–	
$GMAC_{15}$	Setup for GRXDV from GRXCK	10	–	
$GMAC_{16}$	Hold for GRXDV from GRXCK	10	–	

Downloaded from Arrow.com.

Figure 56-57. GMAC MII Mode Signals



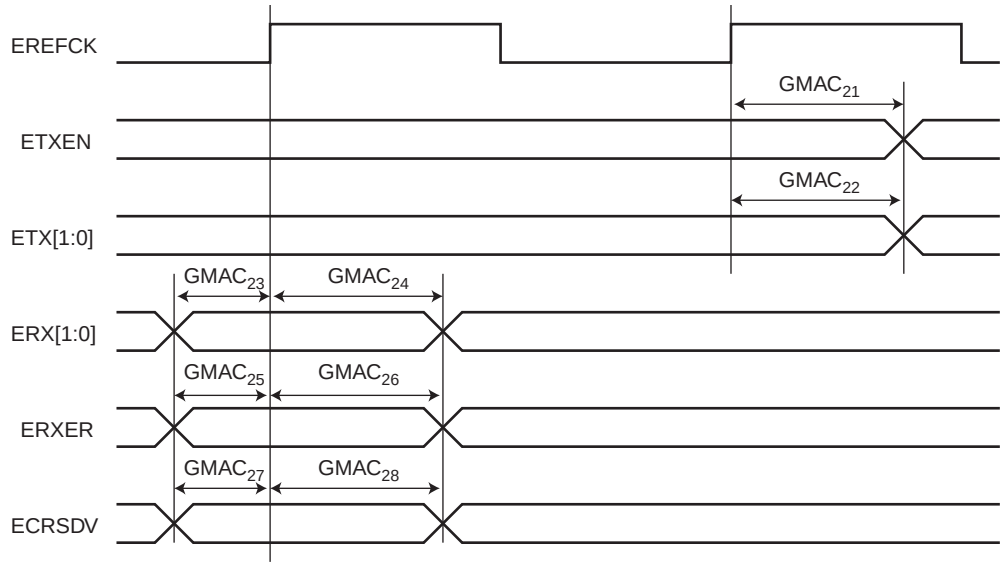
Downloaded from Arrow.com.

RMII Mode

Table 56-91. GMAC RMII Mode Timings

Symbol	Parameter	Min	Max	Unit
$GMAC_{21}$	ETXEN toggling from EREFCK rising	2	16	ns
$GMAC_{22}$	ETX toggling from EREFCK rising	2	16	
$GMAC_{23}$	Setup for ERX from EREFCK rising	4	–	
$GMAC_{24}$	Hold for ERX from EREFCK rising	2	–	
$GMAC_{25}$	Setup for ERXER from EREFCK rising	4	–	
$GMAC_{26}$	Hold for ERXER from EREFCK rising	2	–	
$GMAC_{27}$	Setup for ECRSDV from EREFCK rising	4	–	
$GMAC_{28}$	Hold for ECRSDV from EREFCK rising	2	–	

Figure 56-58. GMAC RMII Mode Signals



Downloaded from Arrow.com.

56.14.1.12 SSC Timings

Timing Conditions

Timings are given assuming the load capacitance in Table 56-92.

Table 56-92. Load Capacitance

Supply	C_L Max
3.3V	30 pF
1.8V	20 pF

Timing Extraction

Figure 56-59. SSC Transmitter, TK and TF in Output

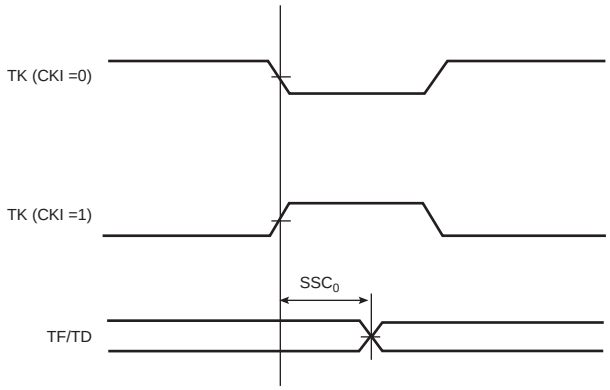
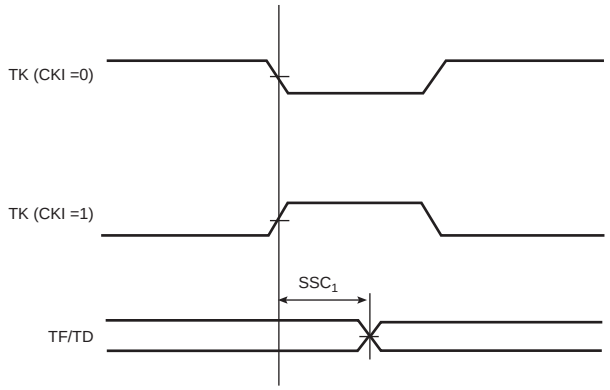


Figure 56-60. SSC Transmitter, TK in Input and TF in Output



Downloaded from Arrow.com.

Figure 56-61. SSC Transmitter, TK in Output and TF in Input

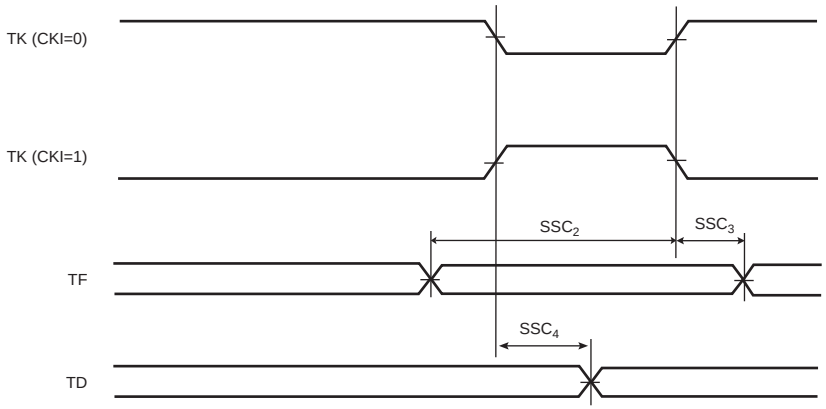


Figure 56-62. SSC Transmitter, TK and TF in Input

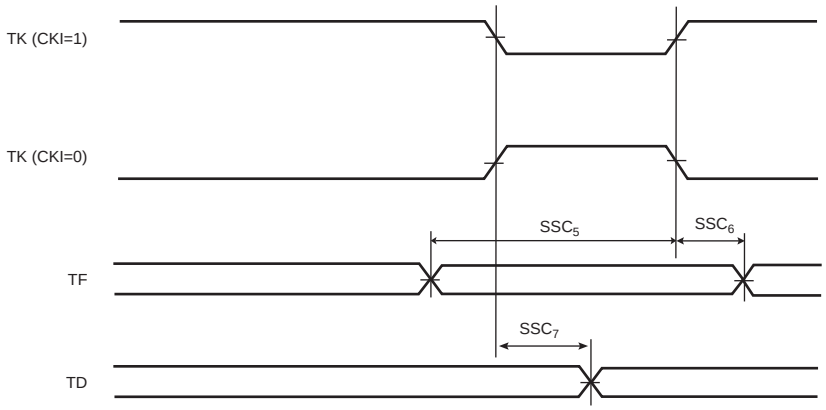
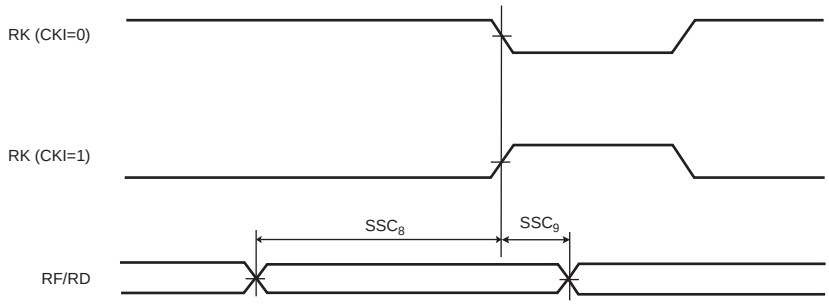


Figure 56-63. SSC Receiver RK and RF in Input



Downloaded from Arrow.com.

Figure 56-64. SSC Receiver, RK in Input and RF in Output

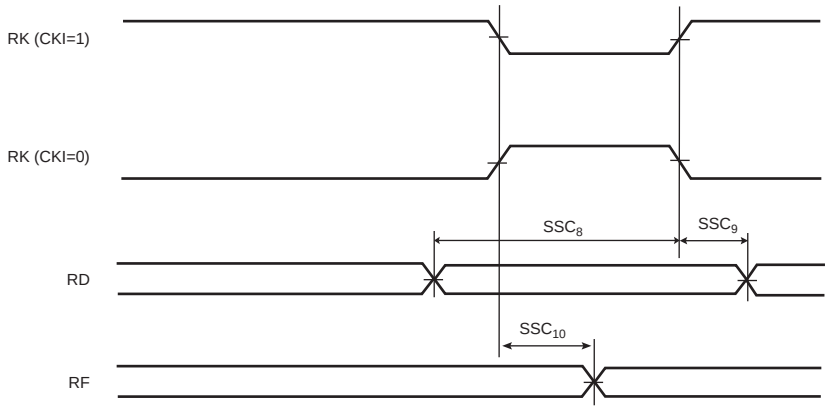


Figure 56-65. SSC Receiver, RK and RF in Output

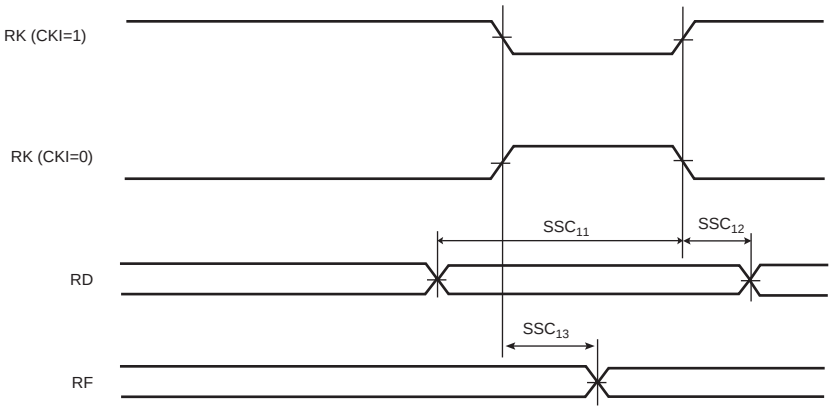
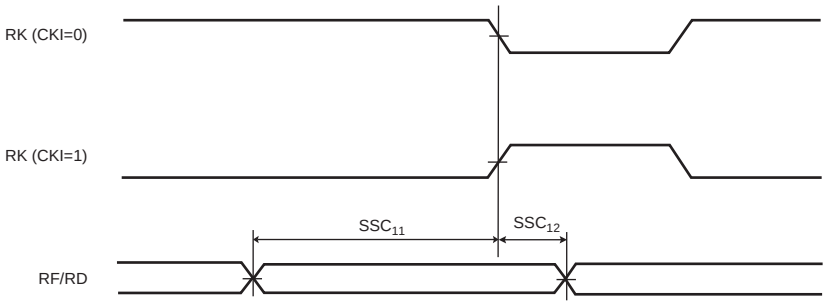


Figure 56-66. SSC Receiver, RK in Output and RF in Input



Downloaded from Arrow.com.

Table 56-93. SSC Timings with 3.3V Peripheral Supply

Symbol	Parameter	Condition	Min	Max	Unit
Transmitter					
SSC_0	TK edge to TF/TD (TK output, TF output)	–	-3.9(1)	3.5(1)	ns
SSC_1	TK edge to TF/TD (TK input, TF output)	–	3.1(1)	11.1(1)	ns
SSC_2	TF setup time before TK edge (TK output)	–	12.0	–	ns
SSC_3	TF hold time after TK edge (TK output)	–	0	–	ns
SSC_4	TK edge to TF/TD (TK output, TF input)	–	-3.9(1)	2.8 (1)	ns
		STTDLY = 0 START = 4, 5 or 7	-3.9 + (2 × t_{CPMCK})(1)	2.8 + (2 × t_{CPMCK})(1)	
SSC_5	TF setup time before TK edge (TK input)	–	0	–	ns
SSC_6	TF hold time after TK edge (TK input)	–	t_{CPMCK}	–	ns
SSC_7	TK edge to TF/TD (TK input, TF input)	–	3.1(1)	10.4(1)	ns
		STTDLY = 0 START = 4, 5 or 7	3.1 + (3 × t_{CPMCK})(1)	10.4 + (3 × t_{CPMCK})(1)	
Receiver					
SSC_8	RF/RD setup time before RK edge (RK input)	–	0	–	ns
SSC_9	RF/RD hold time after RK edge (RK input)	–	t_{CPMCK}	–	ns
SSC_{10}	RK edge to RF (RK input)	–	2.9(1)	8.1(1)	ns
SSC_{11}	RF/RD setup time before RK edge (RK output)	–	8.9 - t_{CPMCK}	–	ns
SSC_{12}	RF/RD hold time after RK edge (RK output)	–	t_{CPMCK} - 2.3	–	ns
SSC_{13}	RK edge to RF (RK output)	–	-2.1(1)	1.9(1)	ns

Note: 1. For output signals (TF, TD, RF), min and max access times are defined. The min access time is the time between the TK (or RK) edge and the signal change. The max access timing is the time between the TK edge and the signal stabilization. Figure 56-67 illustrates min and max accesses for SSC0. The same applies for SSC1, SSC4, and SSC7, SSC10 and SSC13.

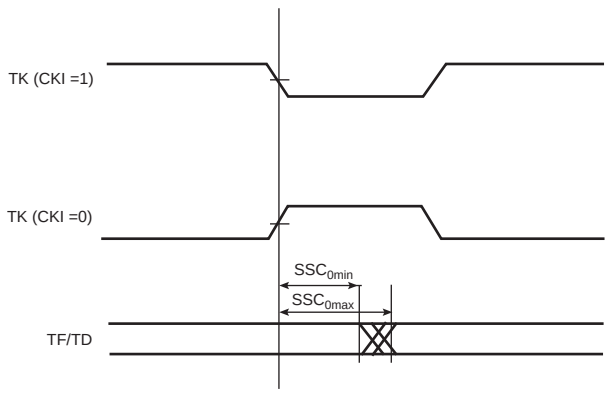
Downloaded from Arrow.com.

Table 56-94. SSC Timings with 1.8V Peripheral Supply

Symbol	Parameter	Condition	Min	Max	Unit
Transmitter					
SSC_0	TK edge to TF/TD (TK output, TF output)	–	-5.9(1)	4.8(1)	ns
SSC_1	TK edge to TF/TD (TK input, TF output)	–	3.6(1)	14.9(1)	ns
SSC_2	TF setup time before TK edge (TK output)	–	15.5	–	ns
SSC_3	TF hold time after TK edge (TK output)	–	0	–	ns
SSC_4	TK edge to TF/TD (TK output, TF input)	–	-5.9(1)	2.9(1)	ns
		STTDLY = 0 START = 4, 5 or 7	-5.9 + (2 × t_{CPMCK})(1)	2.9 + (2 × t_{CPMCK})(1)	
SSC_5	TF setup time before TK edge (TK input)	–	0	–	ns
SSC_6	TF hold time after TK edge (TK input)	–	t_{CPMCK}	–	ns
SSC_7	TK edge to TF/TD (TK input, TF input)	–	3.6(1)	13.0(1)	ns
		STTDLY = 0 START = 4, 5 or 7	3.6 + (3 × t_{CPMCK})(1)	13.0 + (3 × t_{CPMCK})(1)	
Receiver					
SSC_8	RF/RD setup time before RK edge (RK input)	–	0	–	ns
SSC_9	RF/RD hold time after RK edge (RK input)	–	t_{CPMCK}	–	ns
SSC_{10}	RK edge to RF (RK input)	–	3.5(1)	11.3(1)	ns
SSC_{11}	RF/RD setup time before RK edge (RK output)	–	12.3 - t_{CPMCK}	–	ns
SSC_{12}	RF/RD hold time after RK edge (RK output)	–	t_{CPMCK} - 2.9	–	ns
SSC_{13}	RK edge to RF (RK output)	–	-2.9(1)	2.6(1)	ns

Notes: 1. For output signals (TF, TD, RF), min and max access times are defined. The min access time is the time between the TK (or RK) edge and the signal change. The max access timing is the time between the TK edge and the signal stabilization. Figure 56-67 illustrates min and max accesses for SSC0. The same applies for SSC1, SSC4, and SSC7, SSC10 and SSC13.

Figure 56-67. Min and Max Access Time of Output Signals



Downloaded from Arrow.com.

56.14.1.13 ISI Timings

Timing Conditions

Timings are given assuming the load capacitance in Table 56-95.

Table 56-95. Load Capacitance

Supply	C_L Max
3.3V	30 pF
1.8V	20 pF

Timing Extraction

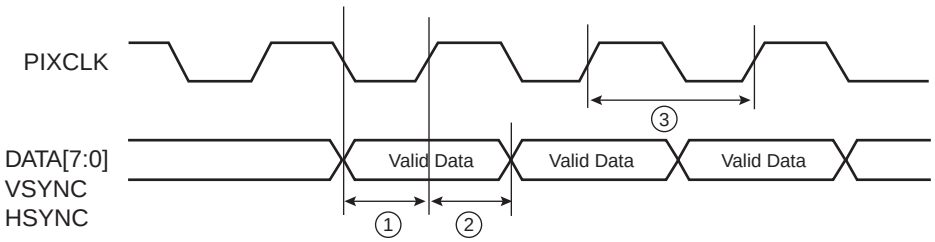
Table 56-96. ISI Timings with Peripheral Supply 3.3V

Symbol	Parameter	Min	Max	Unit
ISI_1	DATA/VSYNC/HSYNC setup time	1.2	–	ns
ISI_2	DATA/VSYNC/HSYNC hold time	-1.0	–	ns
ISI_3	PIXCLK frequency	–	75	MHz

Table 56-97. ISI Timings with Peripheral Supply 1.8V

Symbol	Parameter	Min	Max	Unit
ISI_1	DATA/VSYNC/HSYNC setup time	1.8	–	ns
ISI_2	DATA/VSYNC/HSYNC hold time	-1.4	–	ns
ISI_3	PIXCLK frequency	–	75	MHz

Figure 56-68. ISI Timing Diagram

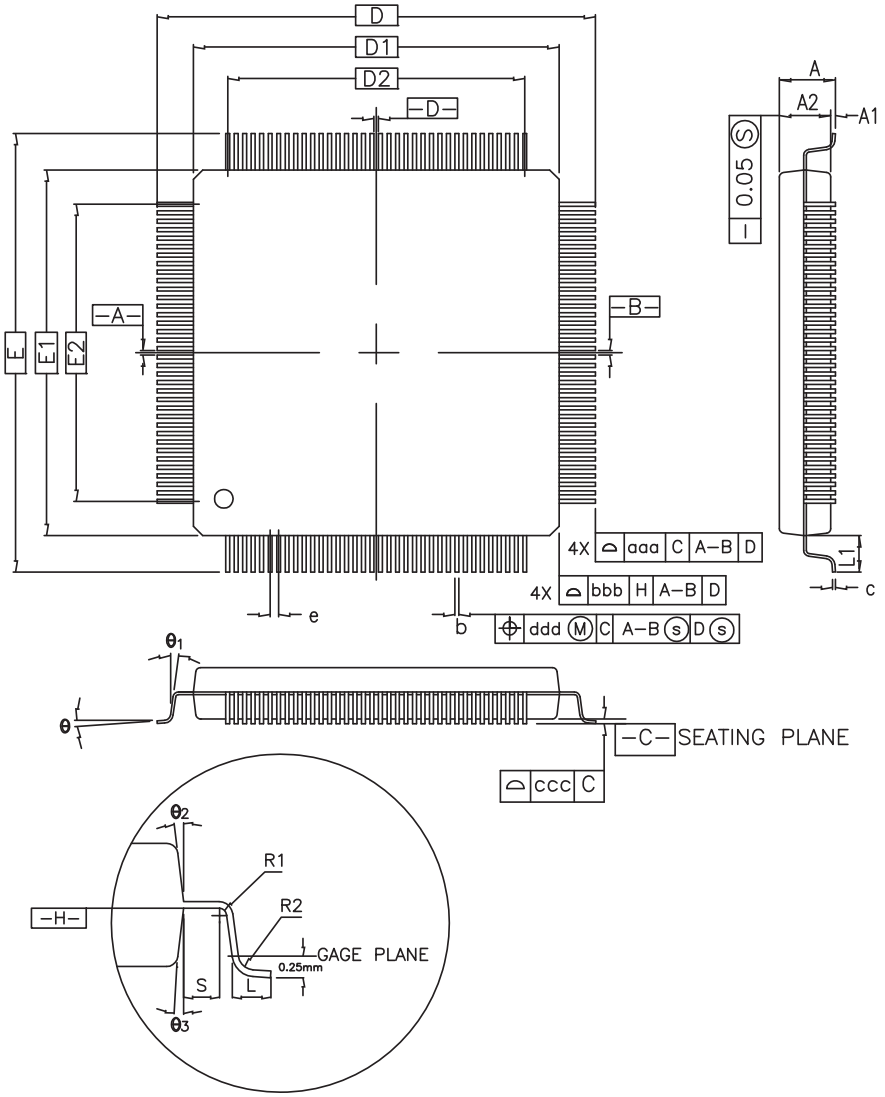


Downloaded from Arrow.com.

57. Mechanical Characteristics

57.1 144-pin LQFP Package

Figure 57-1. 144-pin LQFP Package Mechanical Drawing



CONTROL DIMENSIONS ARE IN MILLIMETERS.

SYMBOL	MILLIMETER			INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	—	—	1.60	—	—	0.063
A1	0.05	—	0.15	0.002	—	0.006
A2	1.35	1.40	1.45	0.053	0.055	0.057
D	22.00 BSC.			0.866 BSC.		
D1	20.00 BSC.			0.787 BSC.		
E	22.00 BSC.			0.866 BSC.		
E1	20.00 BSC.			0.787 BSC.		
R2	0.08	—	0.20	0.003	—	0.008
R1	0.08	—	—	0.003	—	—
θ	0°	3.5°	7°	0°	3.5°	7°
θ1	0°	—	—	0°	—	—
θ2	11°	12°	13°	11°	12°	13°
θ3	11°	12°	13°	11°	12°	13°
c	0.09	—	0.20	0.004	—	0.008
L	0.45	0.60	0.75	0.018	0.024	0.030
L1	1.00 REF			0.039 REF		
S	0.20	—	—	0.008	—	—

SYMBOL	144L					
	MILLIMETER			INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
b	0.17	0.20	0.27	0.007	0.008	0.011
e	0.50 BSC.			0.020 BSC.		
D2	17.50			0.689		
E2	17.50			0.689		
TOLERANCES OF FORM AND POSITION						
aaa	0.20			0.008		
bbb	0.20			0.008		
ccc	0.08			0.003		
ddd	0.08			0.003		

Table 57-1. Device and LQFP Package Maximum Weight

1365	mg

Table 57-2. LQFP Package Reference

JEDEC Drawing Reference	JEDEC
JESD97 Classification	e3

Table 57-3. LQFP Package Characteristics

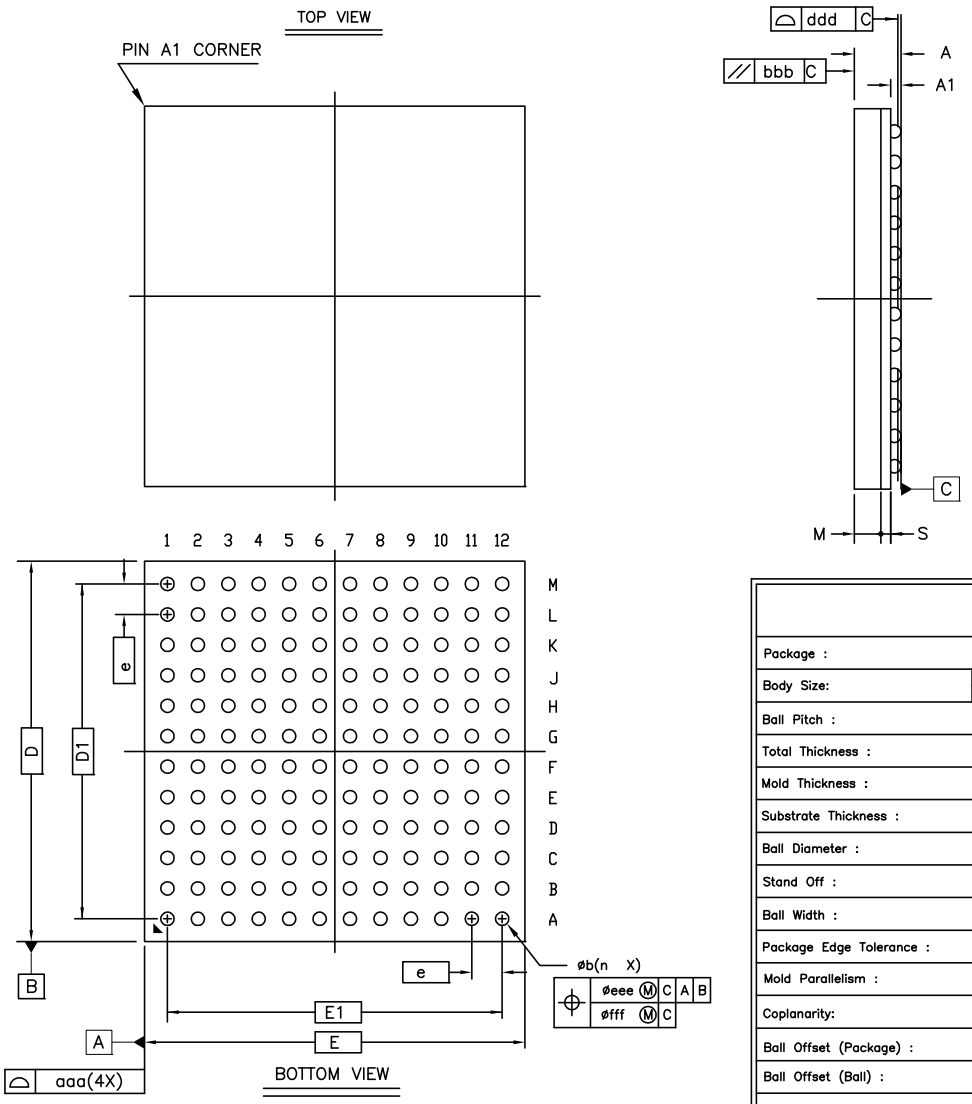
Moisture Sensitivity Level	3

This package respects the recommendations of the NEMI User Group.

Downloaded from Arrow.com.

57.2 144-ball LFBGA Package

Figure 57-2. 144-ball LFBGA Package Mechanical Drawing



		Symbol	Common Dimensions (mm)		
			MIN.	NOM.	MAX.
Package :			LFBGA		
Body Size:	X	E		10.000	
	Y	D		10.000	
Ball Pitch :		e		0.800	
Total Thickness :		A			1.400
Mold Thickness :		M		0.700 Ref.	
Substrate Thickness :		S		0.260 Ref.	
Ball Diameter :				0.350	
Stand Off :		A1	0.220	–	0.320
Ball Width :		b	0.320	–	0.420
Package Edge Tolerance :		aaa		0.150	
Mold Parallelism :		bbb		0.200	
Coplanarity:		ddd		0.080	
Ball Offset (Package) :		eee		0.150	
Ball Offset (Ball) :		fff		0.080	
Ball Count :		n		144	
Edge Ball Center to Center :	X	E1		8.800	
	Y	D1		8.800	

Table 57-4. Device and LFBGA Package Maximum Weight

220	mg

Table 57-5. LFBGA Package Reference

JEDEC Drawing Reference	JEDEC
JESD97 Classification	e8

Table 57-6. LFBGA Package Characteristics

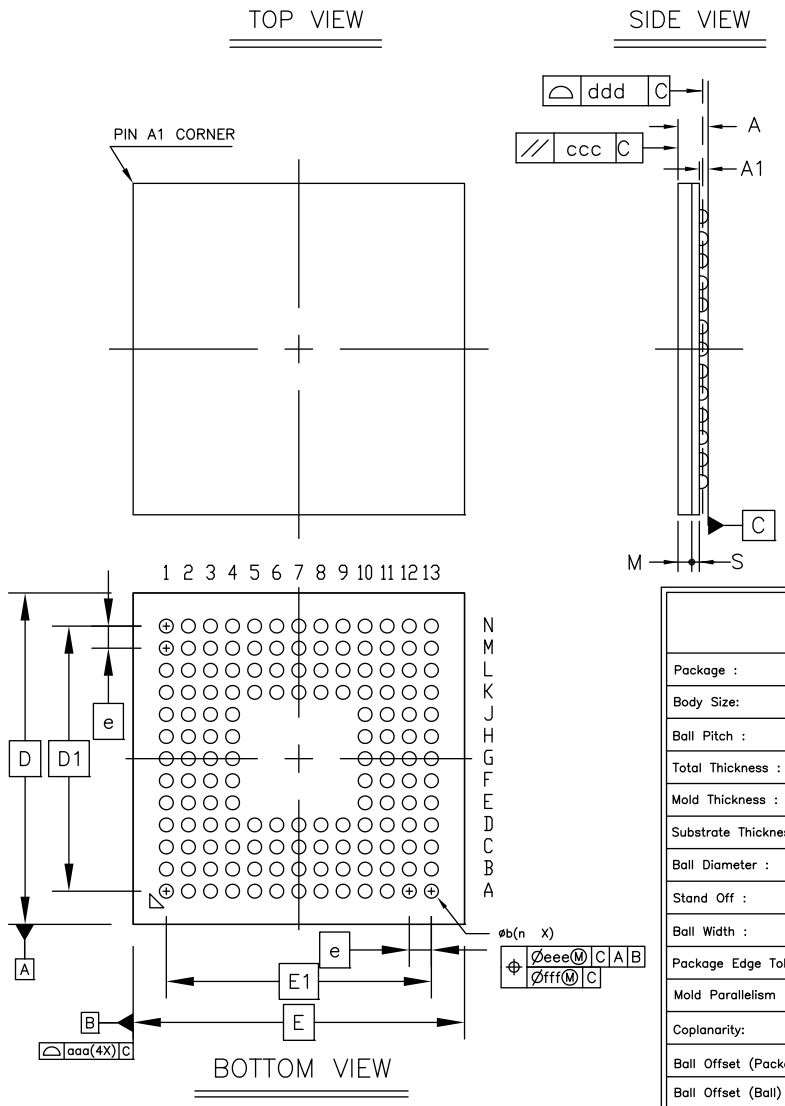
Moisture Sensitivity Level	3

This package respects the recommendations of the NEMI User Group.

Downloaded from Arrow.com.

57.3 144-ball UFBGA Package

Figure 57-3. 144-ball UFBGA Package Mechanical Drawing



		Symbol	Common Dimensions (in mm) MIN.	NOM.	MAX.
Package :			UFBGA		
Body Size:	X	E		6.000	
	Y	D		6.000	
Ball Pitch :		e		0.400	
Total Thickness :		A			0.600
Mold Thickness :		M	0.250	Ref.	
Substrate Thickness :		S	0.136	Ref.	
Ball Diameter :				0.250	
Stand Off :		A1	0.110	–	0.210
Ball Width :		b	0.220	–	0.280
Package Edge Tolerance :		aaa		0.100	
Mold Parallelism :		ccc		0.100	
Coplanarity:		ddd		0.100	
Ball Offset (Package) :		eee		0.150	
Ball Offset (Ball) :		fff		0.050	
Ball Count :		n		144	
Edge Ball Center to Center :	X	E1		4.800	
	Y	D1		4.800	

Table 57-7. Device and UFBGA Package Maximum Weight

36.300	mg

Table 57-8. UFBGA Package Reference

JEDEC Drawing Reference	JEDEC
JESD97 Classification	e3

Table 57-9. UFBGA Package Characteristics

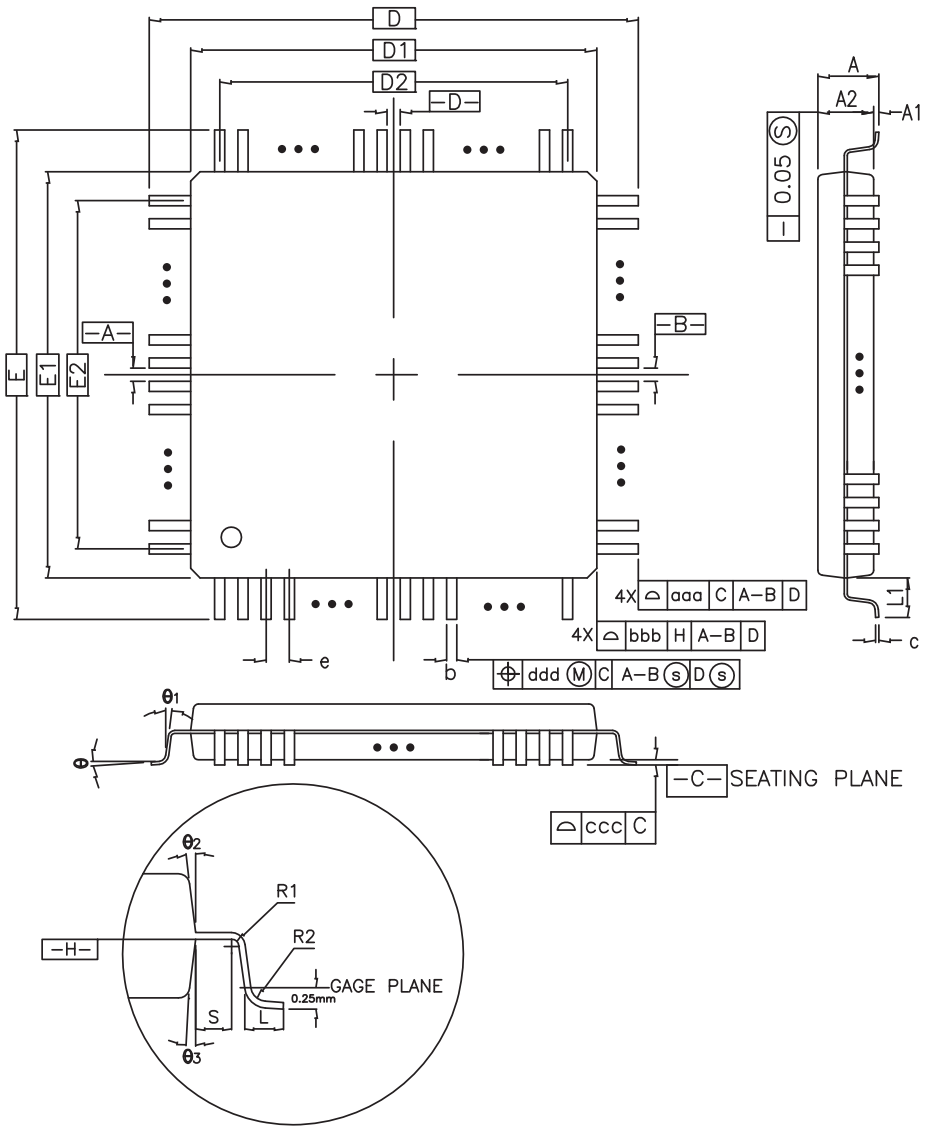
Moisture Sensitivity Level	3

This package respects the recommendations of the NEMI User Group.

Downloaded from Arrow.com.

57.4 100-pin LQFP Package

Figure 57-4. 100-pin LQFP Package Mechanical Drawing



CONTROL DIMENSIONS ARE IN MILLIMETERS.

SYMBOL	MILLIMETER			INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	—	—	1.60	—	—	0.063
A_1	0.05	—	0.15	0.002	—	0.006
A_2	1.35	1.40	1.45	0.053	0.055	0.057
D	16.00 BSC.			0.630 BSC.		
D_1	14.00 BSC.			0.551 BSC.		
E	16.00 BSC.			0.630 BSC.		
E_1	14.00 BSC.			0.551 BSC.		
R_2	0.08	—	0.20	0.003	—	0.008
R_1	0.08	—	—	0.003	—	—
θ	0°	3.5°	7°	0°	3.5°	7°
$θ_1$	0°	—	—	0°	—	—
$θ_2$	11°	12°	13°	11°	12°	13°
$θ_3$	11°	12°	13°	11°	12°	13°
c	0.09	—	0.20	0.004	—	0.008
L	0.45	0.60	0.75	0.018	0.024	0.030
L_1	1.00 REF			0.039 REF		
S	0.20	—	—	0.008	—	—

100L					
MILLIMETER			INCH		
MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
0.17	0.20	0.27	0.007	0.008	0.011
0.50 BSC.			0.020 BSC.		
12.00			0.472		
12.00			0.472		

TOLERANCES OF FORM AND POSITION

0.20	0.008
0.20	0.008
0.08	0.003
0.08	0.003

Table 57-10. Device and LQFP Package Maximum Weight

680	mg

Table 57-11. LQFP Package Reference

JEDEC Drawing Reference	JEDEC
JESD97 Classification	e3

Table 57-12. LQFP Package Characteristics

Moisture Sensitivity Level	3

This package respects the recommendations of the NEMI User Group.

Downloaded from Arrow.com.

57.5 100-ball TFBGA Package

Figure 57-5. 100-ball TFBGA Package Mechanical Drawing

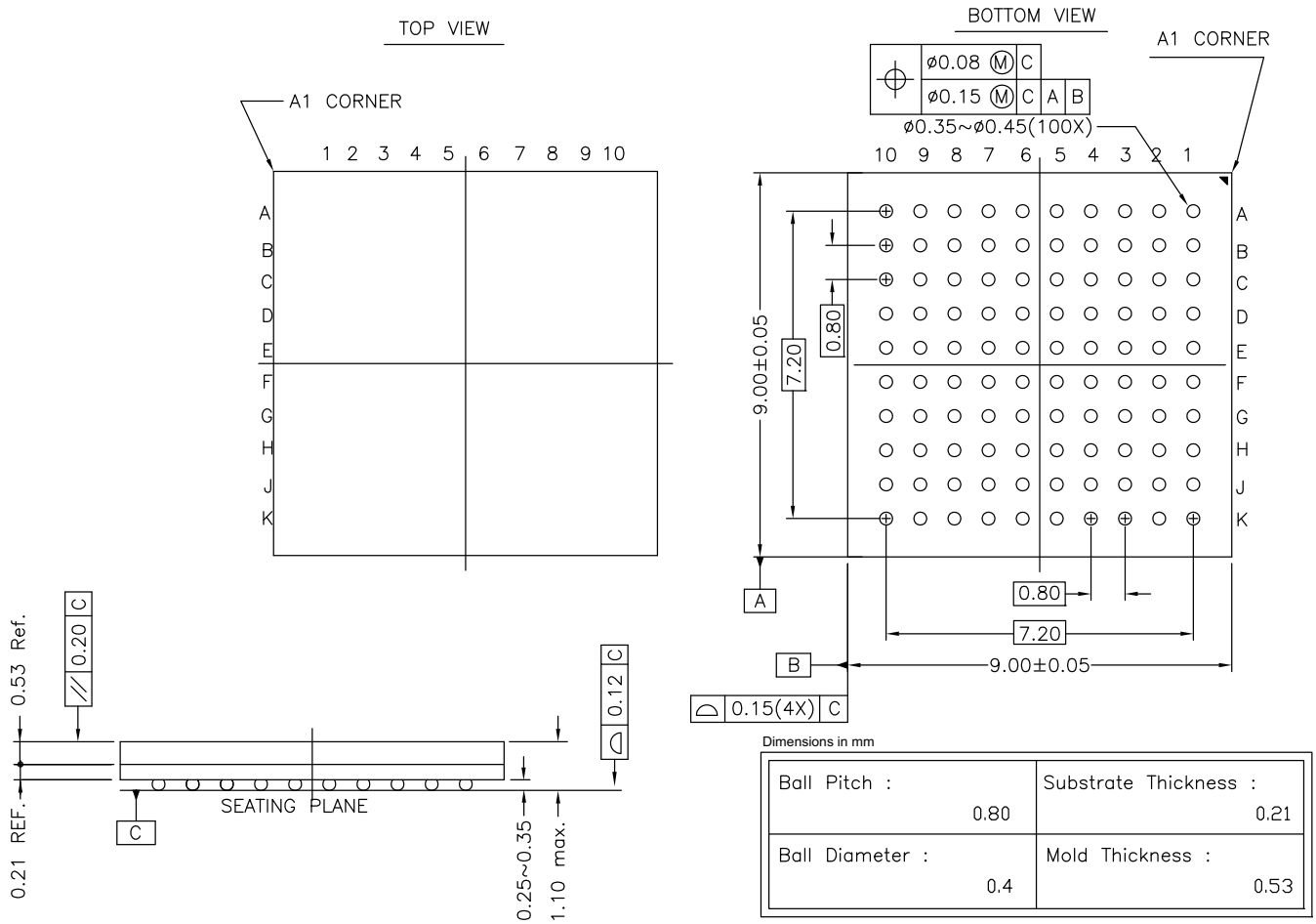


Table 57-13. Device and TFBGA Package Maximum Weight

142	mg

Table 57-14. TFBGA Package Reference

JEDEC Drawing Reference	JEDEC
JESD97 Classification	e8

Table 57-15. TFBGA Package Characteristics

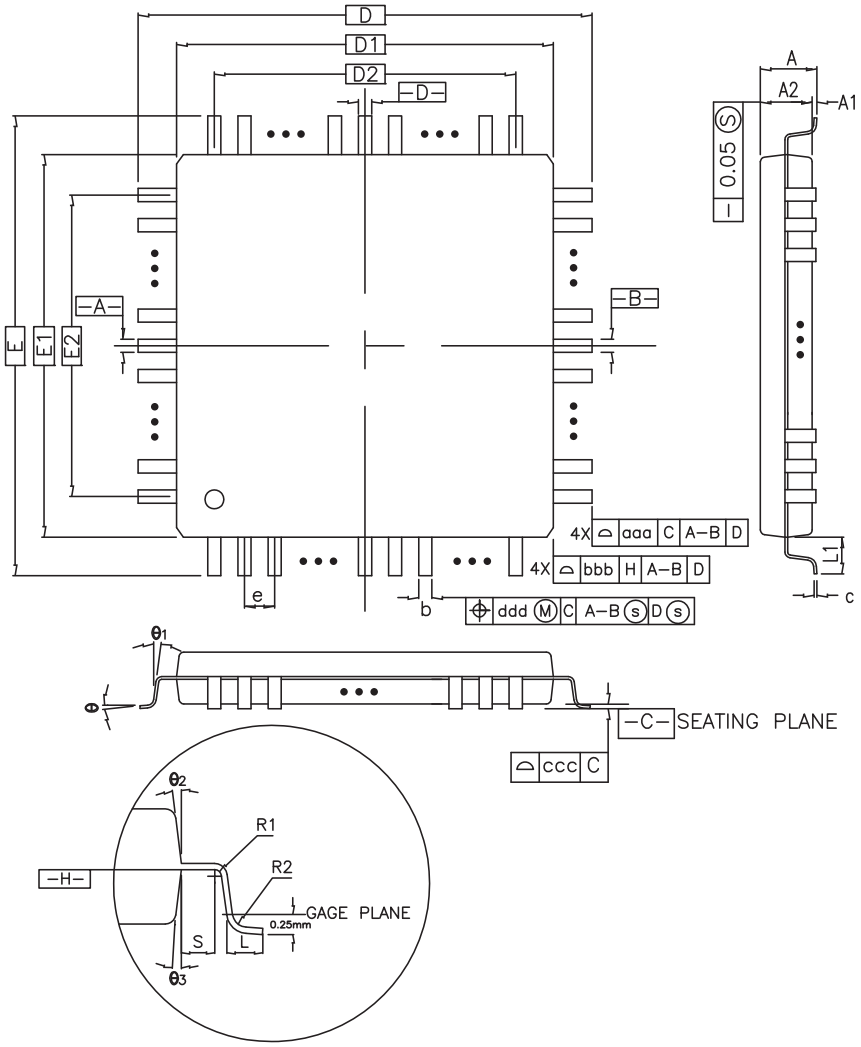
Moisture Sensitivity Level	3

This package respects the recommendations of the NEMI User Group.

Downloaded from Arrow.com.

57.6 64-pin LQFP Package

Figure 57-6. 64-pin LQFP Package Mechanical Drawing



CONTROL DIMENSIONS ARE IN MILLIMETERS.

SYMBOL	MILLIMETER			INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	—	—	1.60	—	—	0.063
A1	0.05	—	0.15	0.002	—	0.006
A2	1.35	1.40	1.45	0.053	0.055	0.057
D	12.00 BSC.			0.472 BSC.		
D1	10.00 BSC.			0.393 BSC.		
E	12.00 BSC.			0.472 BSC.		
E1	10.00 BSC.			0.393 BSC.		
R2	0.08	—	0.20	0.003	—	0.008
R1	0.08	—	—	0.003	—	—
θ	0°	3.5°	7°	0°	3.5°	7°
θ1	0°	—	—	0°	—	—
θ2	11°	12°	13°	11°	12°	13°
θ3	11°	12°	13°	11°	12°	13°
c	0.09	—	0.20	0.004	—	0.008
L	0.45	0.60	0.75	0.018	0.024	0.030
L1	1.00 REF			0.039 REF		
S	0.20	—	—	0.008	—	—

SYMBOL	MILLIMETER			INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
b	0.17	0.20	0.27	0.007	0.008	0.011
e	0.50 BSC.			0.020 BSC.		
D2	7.50			0.295		
E2	7.50			0.295		
TOLERANCES OF FORM AND POSITION						
aaa	0.20			0.008		
bbb	0.20			0.008		
ccc	0.08			0.003		
ddd	0.08			0.003		

Table 57-16. Device and LQFP Package Maximum Weight

370	mg

Table 57-17. LQFP Package Reference

JEDEC Drawing Reference	JEDEC
JESD97 Classification	e3

Table 57-18. LQFP Package Characteristics

Moisture Sensitivity Level	3

This package respects the recommendations of the NEMI User Group.

Downloaded from Arrow.com.

57.7 Soldering Profile

Table 57-19 gives the recommended soldering profile from J-STD-020C.

Table 57-19. Soldering Profile

Profile Feature	Green Package
Average Ramp-up Rate (217°C to Peak)	3°C/sec. max.
Preheat Temperature 175°C ±25°C	180 sec. max.
Temperature Maintained Above 217°C	60 sec. to 150 sec.
Time within 5°C of Actual Peak Temperature	20 sec. to 40 sec.
Peak Temperature Range	260°C
Ramp-down Rate	6°C/sec. max.
Time 25°C to Peak Temperature	8 min. max.

Note: The package is certified to be backward compatible with Pb/Sn soldering profile.

A maximum of three reflow passes is allowed per component.

57.8 Packaging Resources

Land Pattern Definition.

Refer to the following IPC Standards:

- IPC-7351A and IPC-782 (*Generic Requirements for Surface Mount Design and Land Pattern Standards*) http://landpatterns.ipc.org/default.asp
- Atmel Green and RoHS Policy and Package Material Declaration Data Sheet http://www.atmel.com/green/

Downloaded from Arrow.com.

58. Schematic Checklist

The schematic checklist provides the user with the requirements regarding the different pin connections that must be considered before starting any new board design. It also provides information on the minimum hardware resources required to quickly develop an application with the SAM E70 device. It does not consider PCB layout constraints.

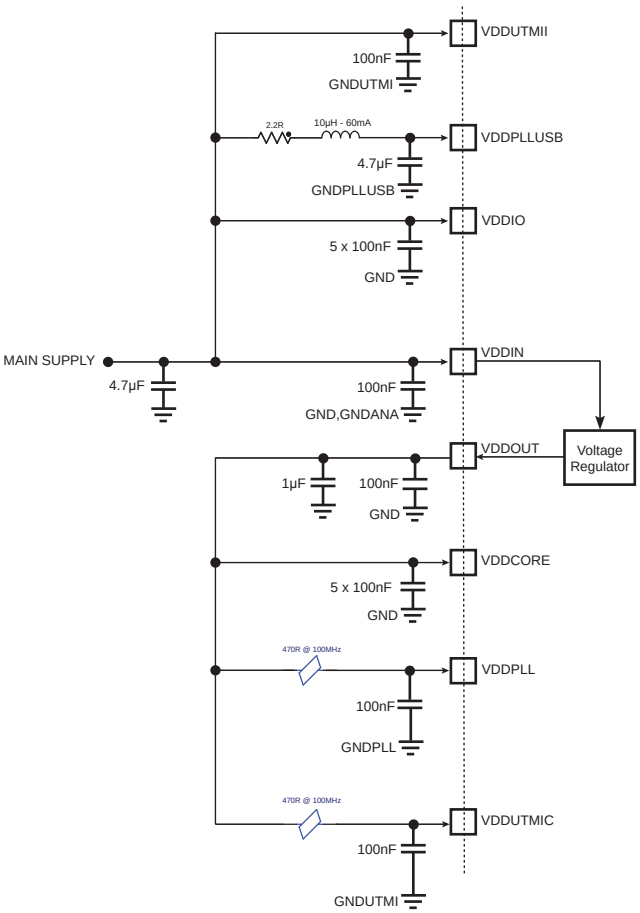
This information is not intended to be exhaustive. Its objective is to cover as many configurations of use as possible. The checklist contains a column for use by designers, making it easy to track and verify each line item.

58.1 Power Supplies

58.1.1 Supplying the device with only one supply

Caution: To guarantee reliable operation of the device, the board design must comply with power-up and power-down sequence guidelines provided in the section “Power Considerations”.

Power Supplies Schematic Example with Internal Regulator Use



Note: Component values are given only as a typical example.

Note: Restrictions
With main supply < 2.5V, USB and DACC are not usable.
With main supply > 2.5V and < 3V, USB is not usable.
With main supply > 3.0 V, all peripherals are usable.

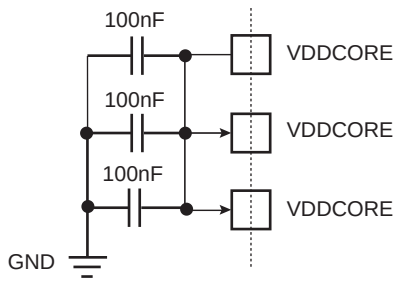
Downloaded from Arrow.com.

☑	Signal Name	Recommended Pin Connection	Description
	VDDIN	Decoupling/filtering capacitors (100 nF and 4.7 µF)[1][2]	Powers the voltage regulator, AFE, DAC, and Analog comparator power supply Supply ripple must not exceed 20 mVrms for 10 kHz to 20 MHz range. **Warning:** VDDIN and VDDIO must have the same level. **Warning:** VDDIN and VDDIO must always be higher than VDDCORE **Warning:** Power-up and power-down sequences given in section “Power Considerations” must be respected.
	VDDIO	Decoupling/filtering capacitors (100 nF)[1][2]	Powers the Peripheral I/O lines (Input/Output Buffers), backup part, 1 Kbyte of Backup SRAM, 32 kHz crystal oscillator, oscillator pads Decoupling/filtering capacitors must be added to improve start-up stability and reduce source voltage drop. Supply ripple must not exceed 30 mVrms for 10 kHz to 10 MHz range. **Warning:** VDDIN and VDDIO must have the same level. **Warning:** VDDIN and VDDIO must always be higher than VDDCORE. **Warning:** Power-up and power-down sequences given in section “Power Considerations” must be respected.
	VDDUTMII	Decoupling capacitor (100 nF)[1][2]	Powers the USB transceiver interface. Must be connected to VDDIO. For USB operations, VDDUTMII and VDDIO voltage ranges must be from 3.0V to 3.6V. Must always be connected even when the USB is not used. Decoupling/filtering capacitors must be added to improve start-up stability and reduce source voltage drop. Supply ripple must not exceed 20 mVrms for 10 kHz to 10 MHz range.
	VDDPLLUSB	Decoupling/filtering RLC circuit[1]	Powers the UTMI PLL and the 3 to 20 MHz oscillator. For USB operations, VDDPLLUSB should be between 3.0V and 3.6V. The VDDPLLUSB power supply pin draws small current, but it is noise sensitive. Care must be taken in VDDPLLUSB power supply routing, decoupling and also on bypass capacitors. Supply ripple must not exceed 10 mVrms for 10 kHz to 10 MHz range.
	VDDOUT	Decoupling capacitor (100 nF + 1 µF)[1][2]	Voltage Regulator Output

Downloaded from Arrow.com.

☑	Signal Name	Recommended Pin Connection	Description
	VDDCORE	Decoupling capacitor (100 nF)(1)(2)	Powers the core, embedded memories and peripherals. Decoupling/filtering capacitors must be added to improve start-up stability and reduce source voltage drop. **Warning:** Power-up and power-down sequences given in section “Power Considerations” must be respected.
	VDDPLL	Decoupling/filtering capacitors ferrite beads (100 nF and 470 Ohm @ 100MHz)(1)(2)	Powers the PLLA and the fast RC oscillator. The VDDPLL power supply pin draws small current, but it is noise sensitive. Care must be taken in VDDPLL power supply routing, decoupling and also on bypass capacitors.
	VDDUTMIC	Decoupling/filtering capacitors ferrite beads (100 nF and 470 Ohm @ 100MHz)(1)(2)	Powers the USB transceiver core. Must always be connected even if the USB is not used. Decoupling/filtering capacitors/ferrite beads must be added to improve start-up stability and reduce source voltage drop.
	GND	Voltage Regulator, Core Chip and Peripheral I/O lines ground	GND pins are common to VDDIN, VDDCORE and VDDIO pins. GND pins should be connected as shortly as possible to the system ground plane.
	GNDUTMI	UDPHS and UHPHS UTMI+ Core and interface ground	GNDUTMI pins are common to VDDUTMII and VDDUTMIC pins. GNDUTMI pins should be connected as shortly as possible to the system ground plane.
	GNDPLL	PLLA cell and Main Oscillator ground	GNDPLL pin is provided for VDDPLL pin. GNDPLL pin should be connected as shortly as possible to the system ground plane.
	GNDANA	Analog ground	GNDANA pins are common to AFE, DAC and ACC supplied by VDDIN pin. GNDANA pins should be connected as shortly as possible to the system ground plane.
	GNDPLLUSB	USB PLL ground	The GNDPLLUSB pin is provided for VDDPLLUSB pin. The GNDPLLUSB pin should be connected as shortly as possible to the system ground plane.

Notes:
1. These values are given only as a typical example.
2. Decoupling capacitors must be connected as close as possible to the microcontroller and on each concerned pin, vias should be avoided.

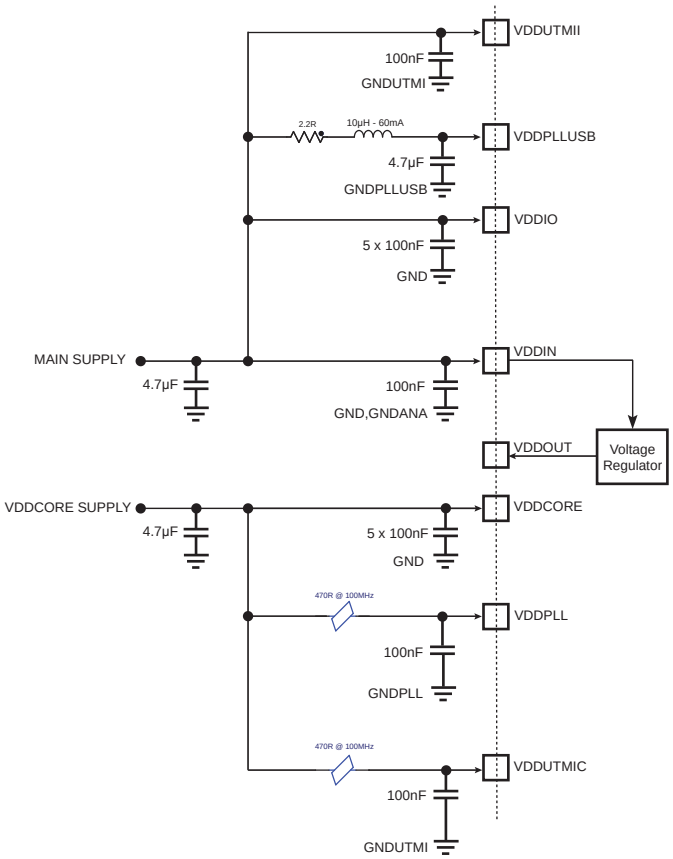


Downloaded from Arrow.com.

58.1.2 Supplying the device with two separate supplies

Caution: The board design must comply with power-up and power down sequence guidelines provided in the section “Power Considerations”.

Power Supplies Schematic Example With Separate Power Supplies[0]



Note: Component values are given only as a typical example

Note: Restrictions
With main supply < 3.0 V, USB is not usable.
With main supply < 2.0 V, AFE, DAC and Analog comparator are not usable.
With main supply and VDDIN > 3.0 V, all peripherals are usable.

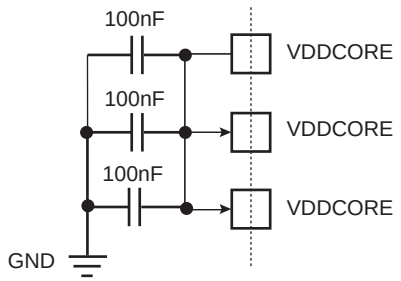
Downloaded from Arrow.com.

☑	Signal Name	Recommended Pin Connection	Description
	VDDIN	Decoupling/filtering capacitors (100 nF and 4.7 µF) (1) (2)	Powers the voltage regulator, AFE, DAC, and Analog comparator power supply Supply ripple must not exceed 20 mVrms for 10 kHz to 20 MHz range. **Warning:** VDDIN and VDDIO must have the same level. **Warning:** VDDIN and VDDIO must always be higher than VDDCORE **Warning:** Power-up and power-down sequences given in Section 6. “Power Considerations” must be respected.
	VDDIO	Decoupling/filtering capacitors (100 nF) (1) (2)	Powers the Peripheral I/O lines (Input/Output Buffers), backup part, 1 Kbytes of Backup SRAM, 32 kHz crystal oscillator, oscillator pads Decoupling/filtering capacitors must be added to improve start-up stability and reduce source voltage drop. Supply ripple must not exceed 30 mVrms for 10 kHz to 10 MHz range. **Warning:** VDDIN and VDDIO must have the same level. **Warning:** VDDIN and VDDIO must always be higher than VDDCORE **Warning:** Power-up and power-down sequences given in section “Power Considerations” must be respected.
	VDDUTMII	Decoupling capacitor (100 nF) (1) (2)	Powers the USB transceiver interface. Must be connected to VDDIO. For USB operations, VDDUTMII and VDDIO voltage ranges must be from 3.0V to 3.6V. Must always be connected even if the USB is not used. Decoupling/filtering capacitors must be added to improve start-up stability and reduce source voltage drop. Supply ripple must not exceed 20 mVrms for 10 kHz to 10 MHz range.
	VDDPLLUSB	Decoupling/filtering RLC circuit (1)	Powers the UTMI PLL and the 3 to 20 MHz oscillator. For USB operations, VDDPLLUSB should be between 3.0V and 3.6V. The VDDPLLUSB power supply pin draws small current, but it is noise sensitive. Care must be taken in VDDPLLUSB power supply routing, decoupling and also on bypass capacitors. Supply ripple must not exceed 10 mVrms for 10 kHz to 10 MHz range.
	VDDOUT	Left unconnected	Voltage Regulator Output
	VDDCORE	Decoupling capacitor (100 nF) (1) (2)	Powers the core, embedded memories and peripherals. Decoupling/filtering capacitors must be added to improve start-up stability and reduce source voltage drop. Supply ripple must not exceed 20 mVrms for 10 kHz to 20 MHz range. **Warning:** Power-up and power-down sequences given in Section 6. “Power Considerations” must be respected.

Downloaded from Arrow.com.

☑	Signal Name	Recommended Pin Connection	Description
	VDDPLL	Decoupling/filtering capacitors ferrite beads (100 nF and 470 Ohm @ 100 MHz) (1) (2)	Powers the PLLA and the fast RC oscillator. The VDDPLL power supply pin draws small current, but it is noise sensitive. Care must be taken in VDDPLL power supply routing, decoupling and also on bypass capacitors. Supply ripple must not exceed 20 mVrms for 10 kHz to 10 MHz range and 10 mVrms for higher frequencies.
	VDDUTMIC	Decoupling/filtering capacitors ferrite beads (100 nF and 470 Ohm @ 100 MHz) (1) (2)	Powers the USB transceiver core. Must always be connected even if the USB is not used. Decoupling/filtering capacitors/ferrite beads must be added to improve start-up stability and reduce source voltage drop. Supply ripple must not exceed 10 mVrms for 10 kHz to 10 MHz range.
	GND	Voltage Regulator, Core Chip and Peripheral I/O lines ground	GND pins are common to VDDIN, VDDCORE and VDDIO pins. GND pins should be connected as shortly as possible to the system ground plane.
	GNDUTMI	UDPHS and UHPHS UTMI+ Core and interface ground	GNDUTMI pins are common to VDDUTMII and VDDUTMIC pins. GNDUTMI pins should be connected as shortly as possible to the system ground plane.
	GNDPLL	PLLA cell and Main Oscillator ground	GNDPLL pin is provided for VDDPLL pin. GNDPLL pin should be connected as shortly as possible to the system ground plane.
	GNDANA	Analog ground	GNDANA pins are common to AFE, DAC and ACC supplied by VDDIN pin. GNDANA pins should be connected as shortly as possible to the system ground plane.
	GNDPLLUSB	USB PLL ground	GNDPLLUSB pin is provided for VDDPLLUSB pin. GNDPLLUSB pin should be connected as shortly as possible to the system ground plane.

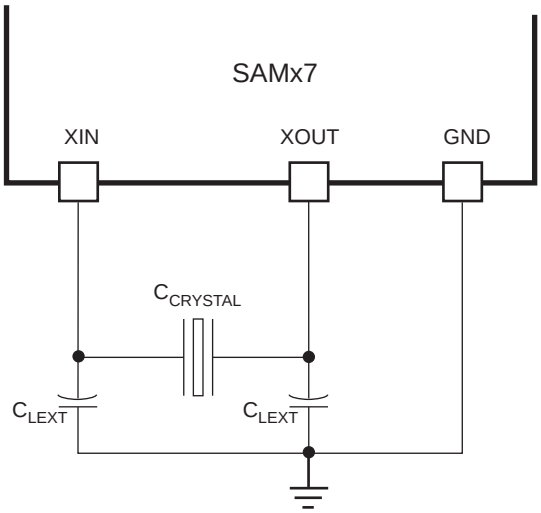
Notes:
1. These values are given only as a typical example.
2. Decoupling capacitors must be connected as close as possible to the microcontroller and on each concerned pin, vias should be avoided.



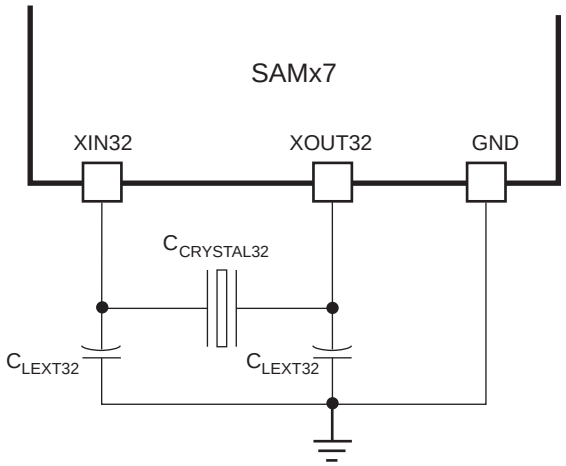
Downloaded from Arrow.com.

58.2 General Hardware Recommendations

58.2.1 Crystal Oscillators

☑	Signal Name	Recommended Pin Connection	Description
	XIN XOUT 3 to 20 MHz Crystal Oscillator in Normal Mode	Crystals between 3 and 20 MHz USB High/Full Speed Host/Device peripherals require a 12 or 16 MHz clock. Capacitors on XIN and XOUT (Crystal Load Capacitance dependent)	Crystal Load Capacitance to check ($C_{CRYSTAL}$).  Example: for a 12 MHz crystal with a load capacitance of $C_{CRYSTAL}$ = 15 pF, external capacitors are required: C_{LEXT} = 12 pF. Refer to Section 56. “Electrical Characteristics”.
	XIN XOUT 3 to 20 MHz Crystal Oscillator in Bypass Mode	XIN: external clock source XOUT: can be left unconnected USB High/Full speed Host/Device peripherals require a 12 or 16 MHz clock.	VDDIO square wave signal External clock source up to 20 MHz Duty Cycle: 40 to 60% Refer to Section 56. “Electrical Characteristics”.
	XIN XOUT 3 to 20 MHz Crystal Oscillator Disabled	XIN: can be left unconnected XOUT: can be left unconnected USB High/Full-speed Host/Device peripherals are not functional with embedded 4/8/12 MHz RC oscillator.	Typical nominal frequency 12 MHz (Internal 4/8/12 MHz RC Oscillator) Duty Cycle: 45 to 55% Refer to Section 56. “Electrical Characteristics”.

Downloaded from Arrow.com.

☑	Signal Name	Recommended Pin Connection	Description
	XIN32 XOUT32 Slow Clock Oscillator	32.768 kHz Crystal Capacitors on XIN32 and XOUT32 (Crystal Load Capacitance dependent)	Crystal load capacitance to check ($C_{CRYSTAL32}$).  Example: for a 32.768 kHz crystal with a load capacitance of $C_{CRYSTAL32}$ = 7 pF, external capacitors are required: C_{LEXT32} = 11 pF. Refer to Section 56. “Electrical Characteristics”.
	XIN32 XOUT32 Slow Clock Oscillator in Bypass Mode	XIN32: external clock source XOUT32: can be left unconnected	VDDIO square wave signal External clock source up to 44 kHz Duty Cycle: 40 to 60% Refer to Section 56. “Electrical Characteristics”.
	XIN32 XOUT32 Slow Clock Oscillator Disabled	XIN32: can be left unconnected XOUT32: can be left unconnected	Typical nominal frequency 32 kHz (internal 32 kHz RC oscillator) Duty Cycle: 45 to 55% Refer to Section 56. “Electrical Characteristics”.

Downloaded from Arrow.com.

58.2.2 Serial Wire Debug Interface

☑	Signal Name	Recommended Pin Connection	Description
	SWCLK/TCK	Pull-up (100 kOhm) (1) If debug mode is not required, this pin can be used as GPIO.	Serial Wire Clock / Test Clock (Boundary scan mode only) This pin is a Schmitt trigger input. No internal pull-up resistor at reset.
	SWDIO/TMS	Pull-up (100 kOhm) (1) If debug mode is not required, this pin can be used as GPIO.	Serial Wire Input-Output / Test Mode Select (Boundary scan mode only). This pin is a Schmitt trigger input. No internal pull-up resistor at reset.
	TDI	Floating. If boundary mode is not required, this pin can be used as GPIO.	Test Data In (Boundary scan mode only) This pin is a Schmitt trigger input. No internal pull-up resistor at reset.
	TRACESWO/TDO	Floating. If debug mode is not required, this pin can be used as GPIO.	Test Data Out (Boundary scan mode only) Output driven at up to VDDIO
	JTAGSEL	In harsh environments(2), it is strongly recommended to tie this pin to GND if not used or to add an external low-value resistor (such as 1 kOhm).	JTAG Selection. Internal permanent pull-down resistor to GNDBU (15 kOhm). Must be tied to VDDIO to enter JTAG Boundary Scan.

Figure 58-1. SWD Schematic Example with a 10-pin Connector

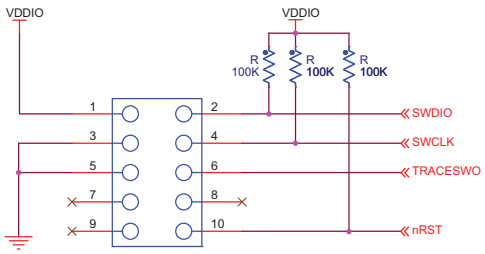
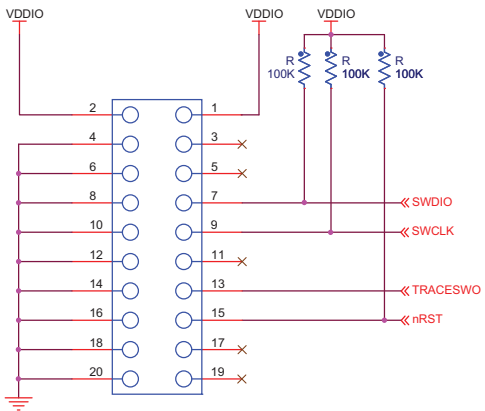


Figure 58-2. SWD Schematic Example with a 20-pin Connector



Notes:
1. These values are given only as a typical example.
2. In a well-shielded environment subject to low magnetic and electric field interference, the pin may be left unconnected. In noisy environments, a connection to ground is recommended.

Downloaded from Arrow.com.

58.2.3 Flash Memory

☑	Signal Name	Recommended Pin Connection	Description
	ERASE	If ERASE mode is not required, this pin can be used as GPIO.	Low level at startup is mandatory when not used.

58.2.4 Reset and Test Pins

☑	Signal Name	Recommended Pin Connection	Description
	NRST	Application dependent. Can be connected to a push button for hardware reset.	NRST is a bidirectional pin (Schmitt trigger input). It is handled by the on-chip reset controller and can be driven low to provide a reset signal to the external components or asserted low externally to reset the microcontroller. By default, the user reset is enabled after a general reset so that it is possible for a component to assert low and reset the microcontroller. A permanent internal pull-up resistor to VDDIO (100 kOhm) is available for user reset and external reset control.
	TST	TST pin can be left unconnected in normal mode. To enter in FFPI mode, TST pin must be tied to VDDIO. In harsh environments (1), it is strongly recommended to tie this pin to GND if not used or to add an external low-value resistor (such as 10 kOhm).	This pin is a Schmitt trigger input. Permanent internal pull-down resistor to GND (15 kOhm).

Note: 1. In a well-shielded environment subject to low magnetic and electric field interference, the pin may be left unconnected. In noisy environments, a connection to ground is recommended.

58.2.5 PIOs

☑	Signal Name	Recommended Pin Connection	Description
	PAx PBx PCx PDx PEx	Application dependent. (Pull-up at VDDIO if needed)	All PIOs are pulled-up inputs (100 kOhm) at reset except those which are multiplexed with Oscillators Drivers and Debug interface that require to be enabled as peripherals: Refer to the column “Reset State” of the pin description tables in Section 5. “Package and Pinout”. Schmitt trigger on all inputs. To reduce power consumption if not used, the concerned PIO can be configured as an output, driven at ‘0’ with internal pull-up disabled.

Downloaded from Arrow.com.

58.2.6 Parallel Capture Mode

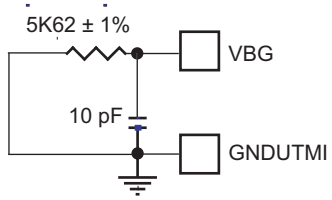
☑	Signal Name	Recommended Pin Connection	Description
	PIODC0–7	Application dependent. (Pull-up at VDDIO)	Parallel mode capture data All are pulled-up inputs (100 kOhm) to VDDIO at reset.
	PIODCCLK	Application dependent. (Pull-up at VDDIO)	Parallel mode capture clock Pulled-up input (100 kOhm) to VDDIO at reset.
	PIODCEN1–2	Application dependent. (Pull-up at VDDIO)	Parallel mode capture mode enable All are pulled-up inputs (100 kOhm) to VDDIO at reset.

58.2.7 Analog Reference, Analog Front-End and DAC

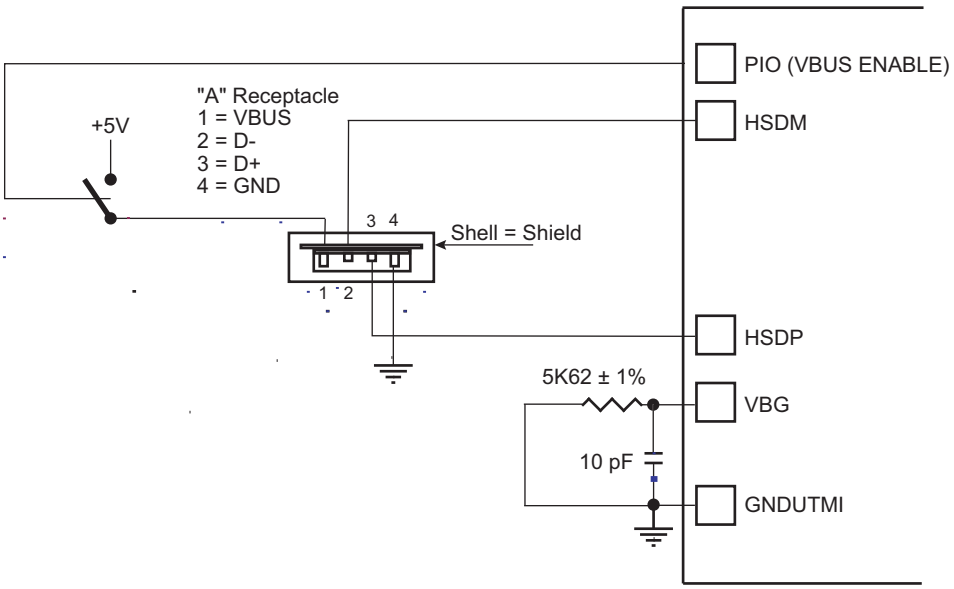
☑	Signal Name	Recommended Pin Connection	Description
Analog Voltage References			
	VREFP	1.7V to VDDIN LC Filter is required.	Positive reference voltage. VREFP is a pure analog input. VREFP is the voltage reference for the AFEC (ADC, PGA DAC and Analog Comparator). To reduce power consumption, if analog features are not used, connect VREFP to GND. Noise must be lower than 100 µVrms
	VREFN	Analog Negative Reference	AFE, DAC and Analog Comparator negative reference VREFN must be connected to GND or GNDANA.
12-bit Analog Front-End			
	AFEx_AD0– AFEx_AD11	0 to VREFP	AFE inputs channels All are pulled-up inputs (100 kOhm) to VDDIO at reset.
	AFEx_ADTRG	Application dependent. (Pulled-up on VREFP)	AFE external trigger input All are pulled-up inputs (100 kOhm) to VDDIO at reset.
12-bit Digital-to-Analog Converter			
	DAC0–DAC1	Application dependent. 0 to VREFP	Analog output All are pulled-up inputs (100 kOhm) to VDDIO at reset.
	DATRG	Application dependent.	DAC external trigger input Pulled-up input (100 kOhm) to VDDIO at reset.

Downloaded from Arrow.com.

58.2.8 USB Host/Device

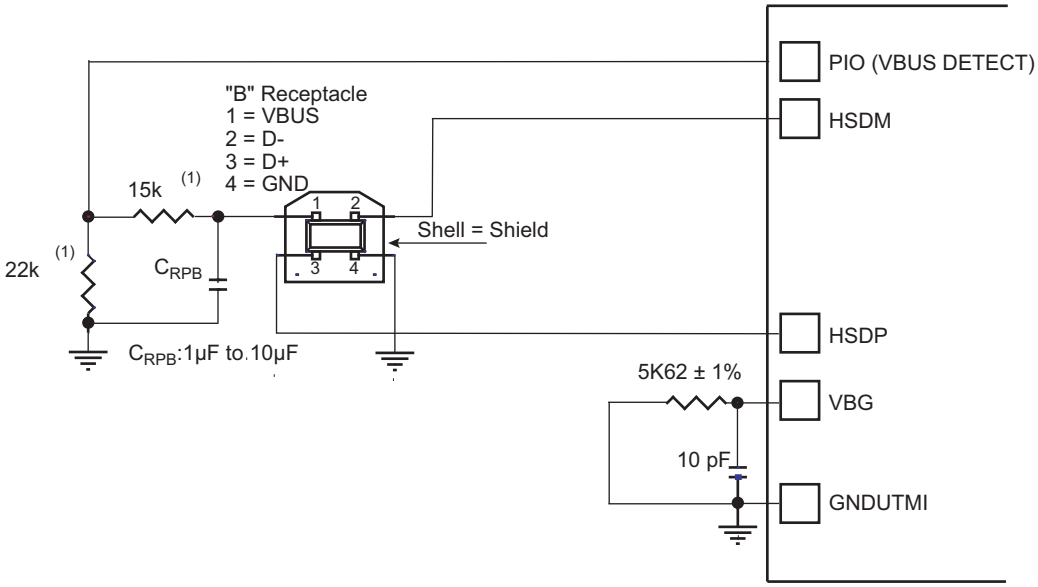
☑	Signal Name	Recommended Pin Connection	Description
	VBG	0.9 - 1.1V (1) (2)	Bias Voltage Reference for USB To reduce the noise on the VBG pin to a minimum, implement the layout considerations below: - Keep the VBG path as short as possible - Ensure a ground connection to GNDUTMI  VBG can be left unconnected if USB is not used.
	HSDM / HSDP	Application dependent (1) (2)	USB High Speed Data Pull-down output at reset.

Notes: 1. The schematic below shows and example of USB High Speed host connection. For more information, refer to Section 37. "USB High-Speed Interface (USBHS)".



Downloaded from Arrow.com.

2. The schematic below shows a typical USB High Speed device connection: For more information, refer to Section 37. “USB High-Speed Interface (USBHS)”.



Note: The values shown on the 22 kΩ and 15 kΩ resistors are only valid with 3.3V supplied PIOs.

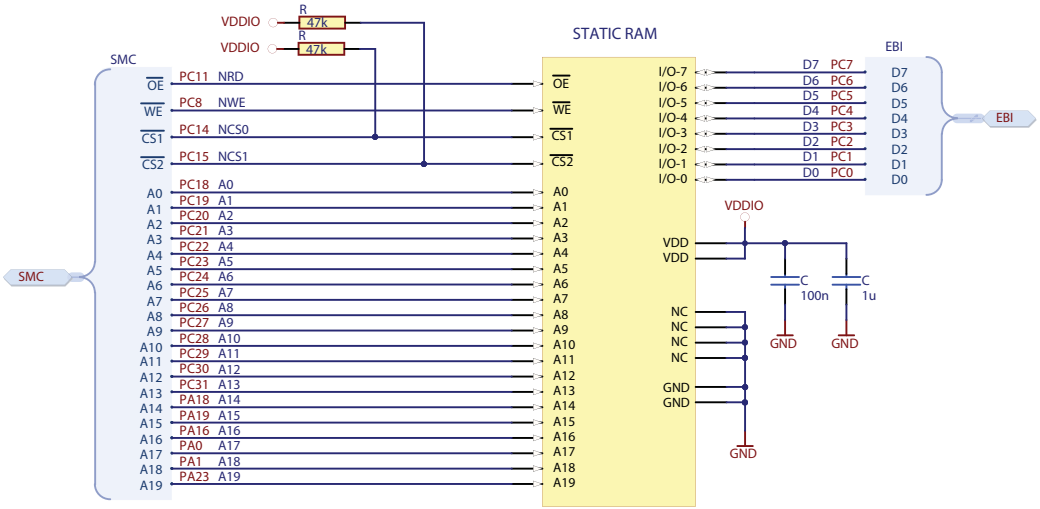
Downloaded from Arrow.com.

58.2.9 Memory Controllers

☑	Signal Name	Recommended Pin Connection	Description
External Bus Interface			
	D[15:0]	Application dependent.	Data Bus (D0 to D15) All data lines are pull-up inputs to VDDIO at reset.
	A[23:0]	Application dependent.	Address Bus (A0 to A23) All address lines pull-up inputs to VDDIO at reset.
	NWAIT	Application dependent.	External Wait Signal. Pulled-up input (100 kOhm) to VDDIO at reset.
Static Memory Controller			
	NCS0-NCS3	Application dependent. (Pull-up at VDDIO)	Chip Select Lines All are pulled-up inputs (100 kOhm) to VDDIO at reset.
	NRD	Application dependent.	Read Signal Pulled-up input (100 kOhm) to VDDIO at reset.
	NWE	Application dependent.	Write Enable All are pulled-up inputs (100 kOhm) to VDDIO at reset.
	NWR0–NWR1	Application dependent.	Write Signals All are pulled-up inputs (100 kOhm) to VDDIO at reset.
	NBS0–NBS1	Application dependent.	Byte Mask Signals All are pulled-up inputs (100 kOhm) to VDDIO at reset.
NAND Flash Logic			
	NANDOE	Application dependent.	NAND Flash Output Enable Pulled-up input (100 kOhm) to VDDIO at reset.
	NANDWE	Application dependent.	NAND Flash Write Enable Pulled-up input (100 kOhm) to VDDIO at reset.
SDR-SDRAM Controller Logic			
	SDCK	Application dependent.	SDRAM Clock Pulled-up input (100 kOhm) to VDDIO at reset.
	SDCKE	Application dependent.	SDRAM Clock Enable Pulled-up input (100 kOhm) to VDDIO at reset.
	SDCS	Application dependent. (Pull-up at VDDIO)	SDRAM Controller Chip Select Pulled-up input (100 kOhm) to VDDIO at reset.
	BA0–BA1	Application dependent.	Bank Select Pulled-up inputs (100 kOhm) to VDDIO at reset.
	SDWE	Application dependent.	SDRAM Write Enable Pulled-up input (100 kOhm) to VDDIO at reset.
	RAS–CAS	Application dependent.	Row and Column Signal Pulled-up inputs (100 kOhm) to VDDIO at reset.
	SDA10	Application dependent.	SDRAM Address 10 Line Pulled-up input (100 kOhm) to VDDIO at reset.

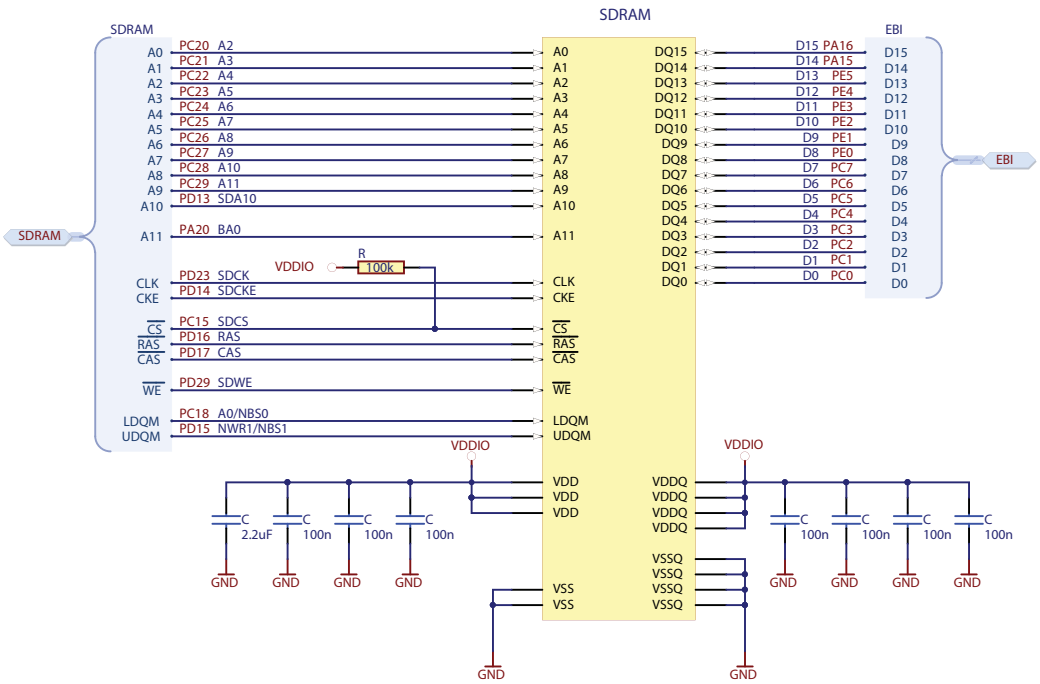
Downloaded from Arrow.com.

Figure 58-3. Schematic Example with a 8 Mb/8-bit Static RAM



Note: For more details on the pin configuration of the EBI, refer to Table 33-3 “EBI Pins and External Static Device Connections“.

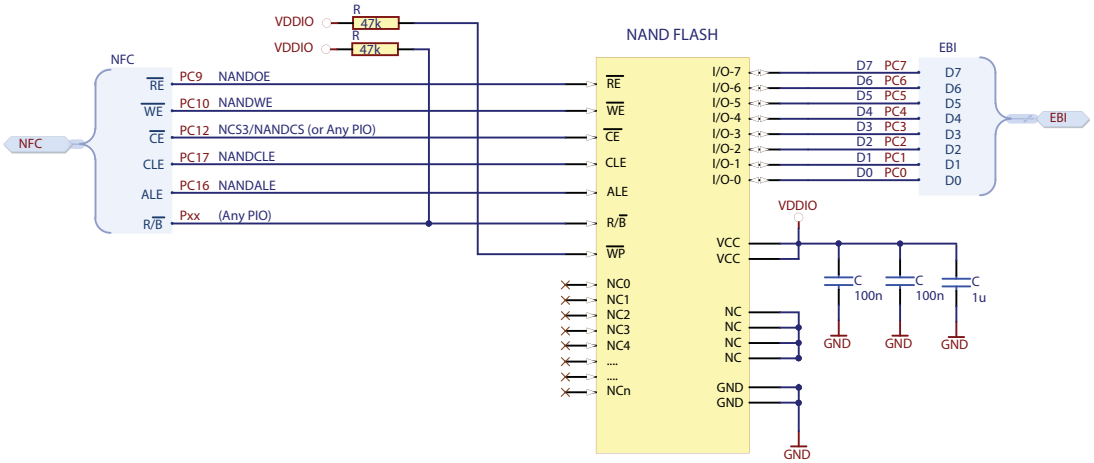
Figure 58-4. Schematic Example with a 16 Mb/16-bit SDRAM



Note: For more details on the pin configuration of the EBI, refer to Table 33-3 “EBI Pins and External Static Device Connections“.

Downloaded from Arrow.com.

Figure 58-5. Schematic Example with a 2 Gb/8-bit NAND Flash



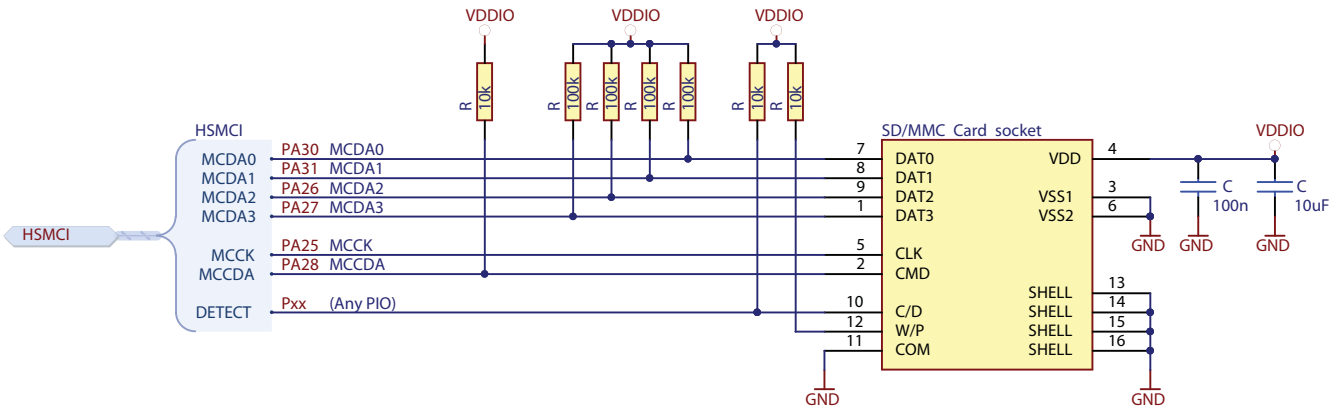
Note: For more details on the pin configuration of the EBI, refer to Table 33-3 “EBI Pins and External Static Device Connections“.

Downloaded from Arrow.com.

58.2.10 High Speed Multimedia Card Interface - HSMCI

☑	Signal Name	Recommended Pin Connection	Description
	MCCK	Application dependent	Multimedia Card Clock Pulled-up input (100 kOhm) to VDDIO at reset.
	MCCDA	Application dependent (Pull-up at VDDIO)	Multimedia Card Slot A Command Pulled-up input (100 kOhm) to VDDIO at reset.
	MCDA0–MCDA3	Application dependent (Pull-up at VDDIO)	Multimedia Card Slot A Data Pulled-up inputs (100 kOhm) to VDDIO at reset.

Figure 58-6. Schematic Example with SD/MMC Card Interface

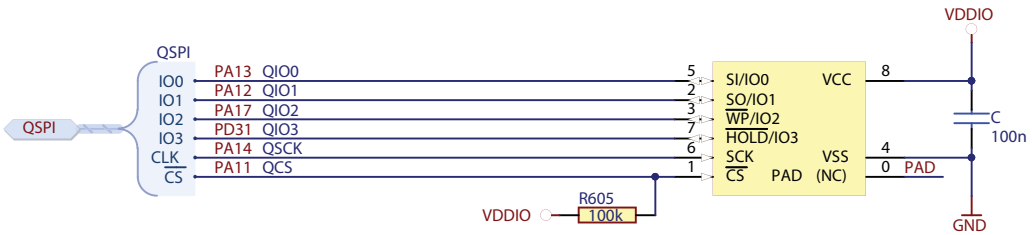


Downloaded from Arrow.com.

58.2.11 QSPI Interface

☑	Signal Name	Recommended Pin Connection	Description
	QSCK	Application dependent.	QSPI Serial Clock Pulled-up input (100 kOhm) to VDDIO at reset.
	QCS	Application dependent. (Pull-up at VDDIO)	QSPI Chip Select Pulled-up input (100 kOhm) to VDDIO at reset.
	QIO0–QIO3	Application dependent.	QSPI I/O Pulled-up inputs (100 kOhm) to VDDIO at reset.

Figure 58-7. Schematic Example with QSPI Data Flash



Downloaded from Arrow.com.

58.2.12 Other Interfaces

☑	Signal Name	Recommended Pin Connection	Description
Universal Synchronous Asynchronous Receiver Transmitter			
	SCKx	Application dependent.	USARTx Serial Clock Pulled-up inputs (100 kOhm) to VDDIO at reset.
	TXDx	Application dependent.	USARTx Transmit Data Pulled-up inputs (100 kOhm) to VDDIO at reset.
	RXDx	Application dependent.	USARTx Receive Data Pulled-up inputs (100 kOhm) to VDDIO at reset.
	RTSx	Application dependent.	USARTx Request To Send Pulled-up inputs (100 kOhm) to VDDIO at reset.
	CTSx	Application dependent.	USARTx Clear To Send Pulled-up inputs (100 kOhm) to VDDIO at reset.
	DTRx	Application dependent.	USARTx Data Terminal Ready Pulled-up inputs (100 kOhm) to VDDIO at reset.
	DSRx	Application dependent.	USARTx Data Set Ready Pulled-up inputs (100 kOhm) to VDDIO at reset.
	DCDx	Application dependent.	USARTx Data Carrier Detect Pulled-up inputs (100 kOhm) to VDDIO at reset.
	RIx	Application dependent.	USARTx Ring Indicator Pulled-up inputs (100 kOhm) to VDDIO at reset.
	LONCOL1	Application dependent.	LON Collision Detection Pulled-up input (100 kOhm) to VDDIO at reset.
Synchronous Serial Controller			
	TD	Application dependent.	SSC Transmit Data Pulled-up input (100 kOhm) to VDDIO at reset.
	RD	Application dependent.	SSC Receive Data Pulled-up input (100 kOhm) to VDDIO at reset.
	TK	Application dependent.	SSC Transmit Clock Pulled-up input (100 kOhm) to VDDIO at reset.
	RK	Application dependent.	SSC Receive Clock I Pulled-up input (100 kOhm) to VDDIO at reset.
	TF	Application dependent.	SSC Transmit Frame Sync Pulled-up input (100 kOhm) to VDDIO at reset.
	RF	Application dependent.	SSC Receive Frame Sync Pulled-up input (100 kOhm) to VDDIO at reset.
Image Sensor Interface			
	ISI_D0–ISI_D11	Application dependent. (Signal can be level-shifted depending on the image sensor characteristics)	Image Sensor Data Pulled-up inputs (100 kOhm) to VDDIO at reset.

Downloaded from Arrow.com.

☑	Signal Name	Recommended Pin Connection	Description
	ISI_MCK	Application dependent. (Signal can be level-shifted depending on the image sensor characteristics)	Image sensor reference clock. No dedicated signal, PCK1 can be used. Pulled-up input (100 kOhm) to VDDIO at reset.
	ISI_HSYNC	Application dependent. (Signal can be level-shifted depending on the image sensor characteristics)	Image sensor horizontal synchro Pulled-up input (100 kOhm) to VDDIO at reset.
	ISI_VSYNC	Application dependent. (Signal can be level-shifted depending on the image sensor characteristics)	Image sensor vertical synchro Pulled-up input (100 kOhm) to VDDIO at reset.
	ISI_PCK	Application dependent. (Signal can be level-shifted depending on the image sensor characteristics)	Image sensor data clock Pulled-up input (100 kOhm) to VDDIO at reset.
Timer/Counter			
	TCLKx	Application dependent.	TC Channel x External Clock Input Pulled-up inputs (100 kOhm) to VDDIO at reset.
	TIOAx	Application dependent.	TC Channel x I/O Line A Pulled-up inputs (100 kOhm) to VDDIO at reset.
	TIOBx	Application dependent.	TC Channel x I/O Line B Pulled-up inputs (100 kOhm) to VDDIO at reset.
Pulse Width Modulation Controller			
	PWMC0_PWMHx PWMC1_PWMHx	Application dependent.	Waveform Output High for Channel x Pulled-up inputs (100 kOhm) to VDDIO at reset.
	PWMC0_PWMLx PWMC1_PWMLx	Application dependent.	Waveform Output Low for Channel x Pulled-up inputs (100 kOhm) to VDDIO at reset.
	PWMC0_PWMFI0– PWMC0_PWMFI2 PWMC1_PWMFI0– PWMC1_PWMFI2	Application dependent.	Fault Inputs Pulled-up inputs (100 kOhm) to VDDIO at reset.
	PWMC0_PWMEXTRG0 PWMC0_PWMEXTRG1 PWMC1_PWMEXTRG0 PWMC1_PWMEXTRG1	Application dependent.	External Trigger Inputs Pulled-up inputs (100 kOhm) to VDDIO at reset.
Serial Peripheral Interface			
	SPIx_MISO	Application dependent.	Master In Slave Out Pulled-up inputs (100 kOhm) to VDDIO at reset.
	SPIx_MOSI	Application dependent.	Master Out Slave In Pulled-up inputs (100 kOhm) to VDDIO at reset.

Downloaded from Arrow.com.

☑	Signal Name	Recommended Pin Connection	Description
	SPIx_SPCK	Application dependent.	SPI Serial Clock Pulled-up inputs (100 kOhm) to VDDIO at reset.
	SPIx_NPCS0	Application dependent. (Pull-up at VDDIO)	SPI Peripheral Chip Select 0 Pulled-up inputs (100 kOhm) to VDDIO at reset.
	SPIx_NPCS1– SPIx_NPCS3	Application dependent. (Pull-up at VDDIO)	SPI Peripheral Chip Select Pulled-up inputs (100 kOhm) to VDDIO at reset.
Two-Wire Interface			
	TWDx	Application dependent. (4.7kOhm Pulled-up on VDDIO)	TWIx Two-wire Serial Data Pulled-up inputs (100 kOhm) to VDDIO at reset.
	TWCKx	Application dependent. (4.7kOhm Pulled-up on VDDIO)	TWIx Two-wire Serial Clock Pulled-up inputs (100 kOhm) to VDDIO at reset.
Fast Flash Programming Interface			
	PGMEN0–PGMEN1	To enter in FFPI mode TST pins must be tied to VDDIO.	Programming Enabling Pulled-up inputs (100 kOhm) to VDDIO at reset.
	PGMM0–PGMM3	Application dependent.	Programming Mode Pulled-up inputs (100 kOhm) to VDDIO at reset.
	PGMD0–PGMD15	Application dependent.	Programming Data Pulled-up inputs (100 kOhm) to VDDIO at reset.
	PGMRDY	Application dependent.	Programming Ready Pulled-up input (100 kOhm) to VDDIO at reset.
	PGMNVALID	Application dependent.	Data Direction Pulled-up input (100 kOhm) to VDDIO at reset.
	PGMNOE	Application dependent.	Programming Read Pulled-up input (100 kOhm) to VDDIO at reset.
	PGMNCMD	Application dependent.	Programming Command Pulled-up input (100 kOhm) to VDDIO at reset.

Downloaded from Arrow.com.

58.3 Boot Program Hardware Constraints

Refer to Section 16. “SAM-BA Boot Program” for more details on the boot program.

58.3.1 Boot Program Supported Crystals (MHz)

A 12 MHz or a 16 MHz crystal or external clock (in Bypass mode) is mandatory in order to generate USB and PLL clocks correctly for the following boots.

58.3.2 SAM-BA Boot

The SAM-BA Boot Assistant supports serial communication via the UART or USB device port:

A. UART0 hardware requirements: none

B. USB Device hardware requirements: external crystal or external clock(1) with a frequency of 12 MHz or 16 MHz

Note: 1. Must be 2500 ppm and VDDIO square wave signal.

Table 58-1. Pins Driven During SAM-BA Boot Program Execution

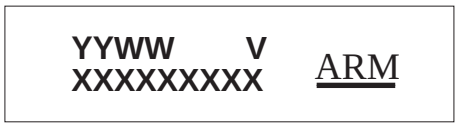
Peripheral	Pin	PIO Line
UART0	URXD0	PA9
UART0	UTXD0	PA10

Downloaded from Arrow.com.

59. Marking

All devices are marked with the Atmel logo and the ordering code.

Additional marking is as follows:



where

- “YY”: Manufactory year
- “WW”: Manufactory week
- “V”: Revision
- “XXXXXXXXX”: Lot number

Downloaded from Arrow.com.

60. Ordering Information

Devices can be ordered in trays or in tape and reel.

Table 60-1 provides ordering codes for tray packing. For tape and reel, append a 'T' to the tray ordering code; e.g. ATSAME70Q21A-CNT.

Table 60-1. Ordering Codes for SAM E70 Devices

Ordering Code	MRL	Flash (Kbytes)	SRAM (Kbytes)	Package	Conditioning	Temperature Operating Range
ATSAME70Q21A-CN	A	2048	384	LFBGA144	Tray	Industrial (-40°C to 105°C)
ATSAME70Q21A-AN		2048	384	LQFP144		
ATSAME70Q21A-CFN		2048	384	UFBGA144(1)		
ATSAME70Q20A-CN		1024	384	LFBGA144		
ATSAME70Q20A-AN		1024	384	LQFP144		
ATSAME70Q20A-CFN		1024	384	UFBGA144(1)		
ATSAME70Q19A-CN		512	256	LFBGA144		
ATSAME70Q19A-AN		512	256	LQFP144		
ATSAME70Q19A-CFN		512	256	UFBGA144(1)		
ATSAME70N21A-CN		2048	384	TFBGA100		
ATSAME70N21A-AN		2048	384	LQFP100		
ATSAME70N20A-CN		1024	384	TFBGA100		
ATSAME70N20A-AN		1024	384	LQFP100		
ATSAME70N19A-CN		512	256	TFBGA100		
ATSAME70N19A-AN		512	256	LQFP100		
ATSAME70J21A-AN		2048	384	LQFP64		
ATSAME70J20A-AN		1024	384	LQFP64		
ATSAME70J19A-AN		512	256	LQFP64		

Note: 1. Contact your local Atmel sales representative for availability.

Downloaded from Arrow.com.

61. Errata

This section contains the known errata found on the SAM E70 parts listed in Section 60. “Ordering Information”.

61.1 AFE Controller (AFEC)

Issue: AFEC_CSELR write protection

The register AFEC_CSELR is not write-protected.

Workaround: None.

Issue: Noise reduces AFEC performance

The AFEC is sensitive to noise. Too much noise may lead to reduced AFEC performances, especially INL, DNL and SNR. The following situations generate noise:

- using a 64-pin QFP package option (it does not have the VREFN pin)
- device activity (e.g. clock tree)
- external components (e.g. missing on-board supply decoupling capacitors)

Note that in the 64-pin QFP package, INL is ±5 LSB, DAC INL is ±3.5 LSB and DAC DNL is ±5 LSB.

Workaround: Adapt the environment to the expected level of performances.

61.2 AHB Peripheral Port (AHBP)

Issue: Access with frequency ratio different from 1 and 1/2 may fail

Peripheral accesses done through the AHBP with a core/bus ratio of 1/3 and 1/4 may lead to unpredictable results.

Workaround: The user must use a core/bus frequency ratio of 1 or 1/2.

61.3 AHB Slave Port (AHBS)

Issue: Latency on accesses with frequency ratio different from 1

DMA accesses done through the AHBS to the TCM with a core/bus ratio of 1/2, 1/3 and 1/4 may lead to latency due to one wait state added to the access from the bus to AHBS.

Workaround: The user must use only the core/bus frequency ratio of 1 to guarantee the length of the access.

Downloaded from Arrow.com.

61.4 ARM Cortex-M7

Issue: **All issues related to the ARM r0p1 core are described on the ARM site**

Workaround: Refer to ARM documentation:

- *ARM Processor Cortex-M7 (AT610) and Cortex-M7 with FPU (AT611) Software Developers Errata Notice*
- *ARM Embedded Trace Macrocell CoreSight ETM–M7 (TM975) Software Developers Errata Notice*

61.5 Extended DMA Controller (XDMAC)

Issue: **Issue with byte and half-word accesses to TCM**

If TCM accesses are generated through the AHBS port of the core, only 32-bit accesses are supported. Accesses which are not 32-bit aligned may overwrite bytes at the beginning and at the end of 32-bit words.

Workaround: The user must use 32-bit aligned buffers and buffers with size of a multiple of 4 bytes when transferring data to or from the TCM through the AHBS port of the core.

Issue: **Issue with byte and half-word accesses with fixed source and fixed destination**

If XDMA is used to transfer 8-bit or 16-bit data in fixed source address or fixed destination address mode, source and destination addresses are incremented by 8-bit or 16-bit.

Workaround: The user can fix the problem by setting the source addressing mode to use microblock and data striding with microblock stride set to 0 and data stride set to -1.

Issue: **DMA request overflow error**

When a DMA memory-to-memory transfer is performed, if the hardware request line selected by the field XDMAC_CCx.PERID toggles when the copy is enabled, the XDMAC_CISx.ROIS flag is set incorrectly.

The memory transfer proceeds normally and the data area is correctly transferred.

Workaround:

Configure the field XDMAC_CCx.PERID to an unused peripheral ID (refer to Table 13-1, “Peripheral Identifiers”).

61.6 Fast Flash Programming Interface (FFPI)

Issue: **FFPI programs only 1 MB of Flash**

The FFPI programs only 1 MB of Flash memory.

Workaround: None.

Downloaded from Arrow.com.

61.7 Inter-IC Sound Controller (I2SC)

Issue: **I2SC not available**

The I2SC is not available.

Workaround: None.

61.8 Master CAN-FD Controller (MCAN)

Issue: **Flexible data rate feature does not support CRC**

CAN-FD peripheral (BOSCH V320) does not support the CRC scheme which includes the stuff bit count introduced by the ISO standardization committee.

CAN 2.0 operation is not impacted.

Workaround: None.

61.9 Power Management Controller (PMC)

Issue: **Wait mode exit from Flash in Standby and Deep-power-down modes may fail**

The delay to exit from Wait mode is too short to respect the Flash wake-up time from Standby and Deep-power-down modes. This may lead to bad opcode fetching.

Workaround:

1. Use the Flash in Idle mode (FLPM = 2).
2. If Flash in Standby mode (FLPM = 0) or in Deep-power-down mode (FLPM = 1) is used, run the wake-up routine from SRAM. This option provides a slight improvement in power consumption.

Downloaded from Arrow.com.

61.10 Quad SPI Interface (QSPI)

Issue: **QSPI hangs with long DLYCS**

QSPI hangs if a command is written to any QSPI register during the DLYCS delay. There is no status bit to flag the end of the delay.

Workaround: The field DLYCS defines a minimum period over which the Chip Select is de-asserted. This is required by some memories. This delay is generally <60ns and comprises internal execution time, arbitration and latencies. Thus, DLYCS must be configured to be slightly higher than the value specified for the slave device. The software must wait for at least this same period of time before a command can be written to the QSPI.

61.11 Serial Synchronous Controller (SSC)

Issue: **Inverted left/right channels**

When the SSC is in Slave mode, the TF signal is derived from the codec and not controlled by the SSC. The SSC transmits the data when detecting the falling edge on the TF signal after the SSC transmission is enabled. In some cases of overflow, a left/right channel inversion may occur.

In this case, the SSC must be re-initialized.

Workaround: Using the SSC in Master mode will ensure that TF is controlled by the SSC. No error occurs. If the SSC must be used in TF Slave mode, the SSC must be started by writing TXEN and RXEN synchronously with TXSYN flag rising in the SSC_SR.

Issue: **Unexpected delay on TD output**

When SSC is configured with the following conditions:

- RCMR.START = Start on falling edge/Start on Rising edge/Start on any edge
- RFMR.FSOS = None (input)
- TCMR.START = Receive Start

an unexpected delay of 2 or 3 system clock cycles is added to TD output.

Workaround: None.

61.12 Supply Controller (SUPC)

Issue: **SUPC_WUIR write protection**

The register SUPC_WUIR of the SUPC is not write-protected.

Workaround: None.

Downloaded from Arrow.com.

61.13 TWI High Speed (TWIHS)

Issue: **TWIHS is not compatible with I2C Hold timing**

The divider to program the hold time is too short to achieve the expected Hold time at high frequency. The achieved time is 227 ns maximum at 150 MHz, instead of the required 300 ns.

Workaround: None.

Issue: **Clear Command does not work**

A bus reset using the CLEAR bit of the TWIHS Control register does not work correctly during a bus busy state.

Workaround: Reconfigure the TWCK line in GPIO output and generate nine clock pulses via software to unlock the I2C device.
Once done, the TWCK line can be reconfigured as a peripheral line.

61.14 Universal Synchronous Asynchronous Receiver Transmitter (USART)

Issue: **Flow Control is not working with DMA**

The RTS signal is not connected to the DMA. Thus, when DMA is used, Flow Control is not supported.

Workaround: None.

61.15 USB High Speed (USBHS)

Issue: **USB Host not working in Low-speed mode**

USB Host does not work in Low-speed mode.

Workaround: None.

Issue: **USB is not working in 64-pin LQFP package**

USB does not work in 64-pin LQFP package option for the following devices:

SAME70J21

SAME70J20

SAME70J19

Workaround: None.

Downloaded from Arrow.com.

Issue: **No DMA for Endpoint 7**

The DMA feature is not available for Pipe/Endpoint 7.

Workaround: None.

Downloaded from Arrow.com.

62. Revision History

Table 62-1. SAM E70 Datasheet Rev. 11296D – Revision History

Date	Changes
19-Jan-16	Added UFBGA144 package to features, configuratiaon summary, package and pinout, mechanical drawings, ordering information and AMR.
	"Features": corrected number of I/O lines. Added I2SC. Table 2-1 "Configuration Summary":updated table. Added I2SC. Table 4-1 "Signal Description List": added I2SC. Removed redundant content from column "Comments". Section 5. "Package and Pinout": added information on reset state in pinout tables. Table 5-1 "144-lead Package Pinout": changed I/O type for all SDA10 to GPIO_AD. Added I2SC pins. Table 5-2 "100-lead Package Pinout": changed I/O type for all SDCK to GPIO_CLK. Added I2SC pins. Updated Table 6-1 "Power Supplies". Updated Figure 9-1, "SAM E70 Product Mapping" with I2SC. Updated Table 11-1 "Real-time Event Mapping List" with I2SC. Section 12.1 "System Controller and Peripherals Mapping": removed sentence on bit band. Updated Table 13-1 "Peripheral Identifiers" with I2SC.
	Section 15. "Debug and Test Features" Removed references to Embedded Trace Buffer (ETB). Section 15.7.8 "IEEE1149.1 JTAG Boundary Scan": updated conditions to enable JTAG boundary scan.
	Section 17. "Fast Flash Programming Interface (FFPI)" Table 17-1 "Signal Description List": updated XIN information. Deleted comment for XIN. Section 17.3 "Parallel Fast Flash Programming", Figure 17-1, "16-bit Parallel Programming Interface": changed input source for XIN. Section 17.3.3 "Entering Parallel Programming Mode": deleted note on device clocking. Reworded steps 2 and 3.
	Section 18. "Bus Matrix (MATRIX)" In Section 18.4.8 "SMC NAND Flash Chip Select Configuration Register": - added warning to bit description SMC_NFCS1. - changed SDRAMEN bit description and added warning.
	Section 21. "Enhanced Embedded Flash Controller (EEFC)" Section 21.4.3.6 "Calibration Bit": updated oscillators that are calibrated in production.
	Section 23. "Watchdog Timer (WDT)" Section 23.1 "Description": Replaced "Idle mode" with "Sleep mode (Idle mode)". Section 23.4 "Functional Description": replaced "Idle mode" with "Sleep mode" Section 23.4 "Functional Description", Section 23.5.2 "Watchdog Timer Mode Register": modified information on WDDIS bit setting to read "When setting the WDDIS bit, and while it is set, the fields WDV and WDD must not be modified."
	Section 24. "Reinforced Safety Watchdog Timer (RSWDT)" Section 24.5.2 "Reinforced Safety Watchdog Timer Mode Register": bit 14 now reserved.
	Section 25. "Reset Controller (RSTC)" Section 25.4.3.1 "General Reset": removed reference to NRSTB.
	Section 26. "Real-time Clock (RTC)" Figure 26-4, "Calibration Circuitry Waveforms": corrected two instances of "3,906 ms" to "3.906 ms". Table 26-2 "Register Mapping": corrected reset for RTC_CALR. Added offset 0xCC as reserved. Section 26.6.1 "RTC Control Register": updated descriptions of value '0' for bits UPDTIM and UPDCAL.

Downloaded from Arrow.com.

Table 62-1. SAM E70 Datasheet Rev. 11296D – Revision History (Continued)

Date	Changes
19-Jan-16	Section 30. “Clock Generator” Section 30.2 “Embedded Characteristics”: updated bullet on embedded RC oscillator. Figure 30-3, "Main Clock Block Diagram": renamed “3-20 MHz Crystal or Ceramic Resonator Oscillator” to “Main Crystal or Ceramic Resonator Oscillator”. Renamed “3-20 MHz Oscillator Counter” to “Main Oscillator Counter”. Section 30.5.1 “Embedded 4/8/12 MHz RC Oscillator”: changed last paragraph beginning “The user can adjust the value...”. Section 30.5.4 ”Main Clock Source Selection”: added that the RC oscillator must be selected for Wait mode. Section 30.7 ”UTMI Phase Lock Loop Programming”: deleted sentence on crystal requirements for USB.
	Section 31. “Power Management Controller (PMC)” Section 31.1 ”Description”: corrected list of oscillators that can be trimmed by software. Section 31.2 ”Embedded Characteristics”: added bullet on generic clock. Updated figure Figure 31-1, "General Clock Block Diagram": replaced “SysTick” with “External SysTick Clock”. Added GCLKx. Section 31.8 ”Peripheral Clock Controller”: added description of the use of the PMC_PCR register. Added Step 5 and WARNING in Section 31.13 “Fast Startup”. Section 31.17 “Programming Sequence”: in Step 7., modified sub-steps (c) and (e). Table 31-4 “Register Mapping”: modified Reset for PMC_OCR; replaced by note. Section 31.20.3 ”PMC System Clock Status Register”: added HCLKS at bit 0. Section 31.20.26 ”PMC Peripheral Control Register”: added GCLKEN, GCLKDIV, DIV and GCLKCSS bits/fields and descriptions. Corrected maximum PID number to 127.
	Section 33. “External Bus Interface (EBI)” Added NAND Flash support on NCS0/1/2 (was NCS3 only). Figure 33-1, "Organization of the External Bus Interface": Removed DQS from block diagram.
	Section 28. “SDRAM Controller (SDRAMC)” Updated Step 1. and Step 4. to Step 9. in Section 28.5.1 “SDRAM Device Initialization”. Section 28.6.5.1 “Self-refresh Mode”: added Note. Section 28.7.3 “SDRAMC Configuration Register”: corrected CAS field configuration values.
	Section 34. “Static Memory Controller (SMC)” Section 34.7.3 “NAND Flash Support”: removed reference to NCS3. Updated Figure 34-5, "NAND Flash Signal Multiplexing on SMC Pins" and added Note 1 below the figure. Section 34.10 “Scrambling/Unscrambling Function”: added details on access for SMC_KEY1 and SMC_KEY2 registers. In Table 34-10 “Register Mapping” and register table sections: SMC OCMS Mode Register now ”“SMC Off-Chip Memory Scrambling Register”. SMC OCMS Key1 Register now ”“SMC Off-Chip Memory Scrambling Key1 Register”. SMC OCMS Key2 Register now “SMC Off-Chip Memory Scrambling Key2 Register”. Section 34.16.5 “SMC Off-Chip Memory Scrambling Register”: corrected bits 8 to 11 to ‘CSxSE’ (were reserved). Section 34.16.6 “SMC Off-Chip Memory Scrambling Key1 Register” and Section 34.16.7 “SMC Off-Chip Memory Scrambling Key2 Register”: added Note (1) to clarify Write-once access.
	Section 35. “DMA Controller (XDMAC)” Updated TC peripheral names and added I2SC in Table 35-1 “Peripheral Hardware Requests”. Table 35-3 “Register Mapping“: corrected access of XDMAC_GTYPE, XDMAC_GWAC, XDMAC_CIM. Section 35.9.6 “XDMAC Global Interrupt Mask Register”: corrected access to Read-only. Section 35.9.28 “XDMAC Channel x [x = 0..23] Configuration Register”: modified INITD bit description.

Downloaded from Arrow.com.

Table 62-1. SAM E70 Datasheet Rev. 11296D – Revision History (Continued)

Date	Changes
19-Jan-16	Section 37. “USB High-Speed Interface (USBHS)” Table 37-1 “Description of USB Pipes/Endpoints”: corrected value in ‘High Bandwidth’ column for Pipe/Endpoint 1. Updated Section 37.4.1 “I/O Lines”. Updated Section 37.5.3.3 “Device Detection” and added Note on VBUS supply. Section 37.6.1 “General Control Register”: changed bit description for VBUSHWC. Section 37.6.2 “General Status Register”: bit 9 now reserved. Section 37.6.3 “General Status Clear Register”: bit 9 now reserved. Section 37.6.4 “General Status Set Register”: changed bit description for VBUSRQS. Section 37.6.12 “Device Endpoint Register”: bit 9 changed from ‘reserved’ to EPEN9. Bit 25 changed from ‘reserved’ to EPRST9. Bits 10 and 11 now reserved in registers: - Section 37.6.6 “Device Global Interrupt Status Register” - Section 37.6.9 “Device Global Interrupt Mask Register” - Section 37.6.10 “Device Global Interrupt Disable Register” - Section 37.6.11 “Device Global Interrupt Enable Register” - Section 37.6.32 “Host Global Interrupt Status Register” - Section 37.6.35 “Host Global Interrupt Mask Register” - Section 37.6.36 “Host Global Interrupt Disable Register” - Section 37.6.37 “Host Global Interrupt Enable Register”
	Section 38. “Ethernet MAC (GMAC)” Updated Section 38.1 “Description”. Section 38.5.2 “Power Management”: deleted reference to PMC_PCER. Section 38.5.3 “Interrupt Sources”: deleted reference to ‘Advanced Interrupt Controller’. Replaced by ‘interrupt controller’. Added information on interrupt sources and priority queues. Section 38.6.14 “IEEE 1588 Support”: Removed reference to ‘output pins’ in 2nd paragraph. Deleted reference to GMAC_TSSx. Section 38.6.15 “Time Stamp Unit” added information on GTSUCOMP signal in last paragraph.
	Section 40. “Serial Peripheral Interface (SPI)” Section 40.8.1 “SPI Control Register”: added bits FIFODIS, FIFOEN, RXFCLR, TXFCLR and REQCLR.
	Section 41. “Quad SPI Interface (QSPI)” Section 41.7.5 “QSPI Status Register”: updated RDRF, TDRE, TXEMPTY, and OVRES field descriptions.
	Section 42. “Two-wire Interface (TWIHS)” Section 42.6.5.7 “High-Speed Slave Mode”: updated 11-MHz limit information. Updated Section 42.6.7 “Register Write Protection”. Updated Section 42.7.1 “TWIHS Control Register”: added bit FIFODIS, FIFOEN, LOCKCLR and THRCLR.
	Added Section 44. “Inter-IC Sound Controller (I2SC)”.

Downloaded from Arrow.com.

Table 62-1. SAM E70 Datasheet Rev. 11296D – Revision History (Continued)

Date	Changes
19-Jan-16	Section 45. “Universal Synchronous Asynchronous Receiver Transceiver (USART)” Section 45.6.1 “Baud Rate Generator”: corrected value in "The frequency of the signal provided on SCK must be at least..." Section 45.6.1.2 “Fractional Baud Rate in Asynchronous Mode” and Section 45.7.23 “USART Baud Rate Generator Register”: added warning “When the value of field FP is greater than 0...” Section 45.6.1.3 “Baud Rate in Synchronous Mode or SPI Mode”: corrected external clock frequency. Corrected SCK maximum frequency. Inserted new Figure 45-27, "RTS Line Software Control when USART_MR.USART_MODE = 2". Section 45.6.3.4 “Manchester Decoder”: corrected “MANE flag” with “MANERR” flag. Section 45.6.8.5 “Character Transmission”: added content to 1st paragraph. Corrected occurrences of RTSEN to RCS, RTSDIS to FCS. Section 45.6.9.8 “Slave Node Synchronization”: updated bullet on oversampling. Section 45.7.1 “USART Control Register”: updated RTSDIS bit description. Section 45.7.3 “USART Mode Register”: updated descripiton for row 0xE, SPI_MASTER.
	Section 47. “Controller Area Network (MCAN)” Section 47.5.7.5 “Standard Message ID Filter Element”: updated description of SFID2[5:0]. Section 47.5.7.6 “Extended Message ID Filter Element”: updated description of EFID2[5:0]. Added Section 47.6.1 “MCAN Core Release Register” and Section 47.6.2 “MCAN Endian Register” and updated Table 47-13 “Register Mapping“. Section 47.6.4 “MCAN Fast Bit Timing and Prescaler Register”: updated FSJW, FTSEG2 and FSTEG1 field description: $t_{core\ clock}$ now t_q Section 47.6.8 “MCAN Bit Timing and Prescaler Register”: updated SJW, TSEG2, TSEG1 and BRP field descriptions: $t_{core\ clock}$ now t_q
	Section 49. “Pulse Width Modulation Controller (PWM)” Updated Figure 49-1, "Pulse Width Modulation Controller Block Diagram". Updated Section 49.6.2.2 “Comparator”. Updated Figure 49-33, "Leading-Edge Blanking". Section 49.6.6.1 “Initialization”: modified “Enable of the interrupts...” list item. Added Section 49.6.6.4 “Changing the Update Period of Synchronous Channels”, Section 49.6.6.5 “Changing the Comparison Value and the Comparison Configuration” and Section 49.6.6.6 “Interrupt Sources”. Added reference to Section 49.5.4 “Fault Inputs” in register descriptions. Corrected PWM period formulas in Section 49.7.43 “PWM Channel Period Register”and Section 49.7.44 “PWM Channel Period Update Register”. Section 49.7.50 “PWM Leading-Edge Blanking Register”: updated LEBDELAY bit description.
	Section 50. “Analog Front-End Controller (AFEC)” Updated Section 50.6 “Functional Description”. Updated Section 50.6.12 “AFE Timings”. Section 50.6.18 “Register Write Protection”: added “AFEC Channel Differential Register” to the list of write-protected registers. Section 50.7.5 “AFEC Channel Sequence 2 Register”: corrected number of channels to 12. Added sentence on write protection below the register table for: Section 50.7.20 “AFEC Channel Offset Compensation Register” Section 50.7.21 “AFEC Temperature Sensor Mode Register” Section 50.7.25 “AFEC Correction Select Register” Section 50.7.26 “AFEC Correction Values Register” Section 50.7.27 “AFEC Channel Error Correction Register”

Downloaded from Arrow.com.

Table 62-1. SAM E70 Datasheet Rev. 11296D – Revision History (Continued)

Date	Changes
19-Jan-16	Section 51. “Digital-to-Analog Converter Controller (DACC)” External Trigger mode changed to Trigger mode throughout. Removed references to ‘pipelined architecture’ and ‘pipeline’ throughout. Updated Figure 51-1, "Block Diagram". Updated Section 51.6.1 “Digital-to-Analog Conversion”. Added sentence on DACRDY. Changed ‘maximum conversion rate’ to ‘minimum conversion period’. Section 51.6.4.1 “Trigger Mode”: removed fragment ‘(either DATRG pin or timer counter events)’. Section 51.6.4.2 “Free-Running Mode”: added sentence on FIFO. Updated Figure 51-3, "Conversion Sequence in Free-running Mode". Section 51.6.4.3 “Max Speed Mode”: updated paragraph on conversion rate. Added sentence on FIFO. Updated Section 51.6.4.4 “Bypass Mode”. Table 51-4 “Register Mapping”: modified reset value for DACC_MR. Section 51.7.3 “DACC Trigger Register”: bit description changed for TRGSEL bit.
	Section 53. “Integrity Check Monitor (ICM)” Section 53.5.2.2 “ICM Region Configuration Structure Member”: removed MRPROT field. Section 53.6.1 “ICM Configuration Register”: removed fields HAPROT and DAPROT; updated description DUALBUFF field
	Section 58. “Schematic Checklist” Removed redundant voltage ranges for power supplies. Decoupling/filtering capacitors recommendation changed from 10 µF to 4.7 µF. Section 58.2.1 “Crystal Oscillators”: XIN/XOUT: in all modes, updated ‘12 Mhz Main Oscillator’ to 3 to 20 MHz Crystal Oscillator’. XIN/XOUT in Normal and Bypass modes: updated USB clock frequency requirements. XIN/XOUT disabled: updated comment on USB. Updated internal RC oscillator frequency. XIN/XOUT in Bypass mode: updated external clock source frequency.
	Section 56. “Electrical Characteristics” Table 56-1 “Absolute Maximum Ratings*“: added 144-ball UFBGA. Table 56-2 “DC Characteristics“: added max slope for V_{DDIO} and V_{DDCORE}. Updated V_{IL},V_{IH} and R_{SERIAL} parameters. Updated conditions in Section 56.3.6 “I/O Switching Power Consumption”. Section 56.11 “12-bit DAC Characteristics”: in Table 56-49 “Static Performance Characteristics“ updated INL and DNL parameters, updated V_{DDIN}. Table 56-22 “4/8/12 MHz RC Oscillator Characteristics“: updated untrimmed 4 MHz accuracy. Updated Note (3). Table 56-23 “32.768 kHz Crystal Oscillator Characteristics“: updated parameter Startup Time. Table 56-30 “PLLUSB Characteristics“: updated parameter Input Frequency. Added Note (1). Section 56.13 “Timings for Worst-Case Conditions”: deleted redundant Table 56-55 “Worst-case Conditions”. Section 56.13.1.5 “SPI Characteristics”: added information on sampling point of MISO in master mode. Added Figure 56-19, "MISO Capture in Master Mode". In “Master Read Mode”, updated values for a non-volatile memory. Table 56-62 “SMC Write Signals - NWE Controlled (WRITE_MODE = 1)“: updated SMC_{19} and SMC_{21} parameters. Updated conditions in Table 56-77 “Processor Clock Waveform Parameters“ and Table 56-78 “Master Clock Waveform Parameters“. Section 56.13.1.11 ”GMAC Characteristics”: removed note from Table 56-68 “GMAC MII Mode Timings“ and Table 56-69 “GMAC RMII Mode Timings“. Section 56.14.1.5 “SPI Characteristics”: added information on sampling point of MISO in master mode. Added Figure 56-45, "MISO Capture in Master Mode". In “Master Read Mode”, updated values for a non-volatile memory. Table 56-84 “SMC Write Signals - NWE Controlled (WRITE_MODE = 1)“: updated SMC_{19} and SMC_{21} parameters. Section 56.14.1.11 ”GMAC Characteristics”: removed note from Table 56-90 “GMAC MII Mode Timings“ and Table 56-91 “GMAC RMII Mode Timings“.

Downloaded from Arrow.com.

Table 62-1. SAM E70 Datasheet Rev. 11296D – Revision History (Continued)

Date	Changes
19-Jan-16	Section 61. “Errata” Added Section 61.1 “AFE Controller (AFEC)” - “Noise reduces AFEC performance” Section 61.5 “Extended DMA Controller (XDMAC)” - “DMA request overflow error” Section 61.6 “Fast Flash Programming Interface (FFPI)” - “FFPI programs only 1 MB of Flash” Section 61.7 “Inter-IC Sound Controller (I2SC)” - “I2SC not available” Section 61.9 “Power Management Controller (PMC)” - “Wait mode exit from Flash in Standby and Deep-power-down modes may fail” Section 61.10 “Quad SPI Interface (QSPI)” - “QSPI hangs with long DLYCS” Section 61.11 “Serial Synchronous Controller (SSC)” - “Unexpected delay on TD output” Section 61.13 “TWI High Speed (TWIHS)” - “Clear Command does not work” Section 61.15 “USB High Speed (USBHS)” - “No DMA for Endpoint 7” Updated - Section 61.15 “USB High Speed (USBHS)”: “USB is not working in 64-pin LQFP package” Deleted Section Digital-to-Analog Controller (DACC): - “Expected performances are not achieved on the full power supply range”

Downloaded from Arrow.com.

Table 62-2. SAM E70 Datasheet Rev. 11296C – Revision History

Date	Changes
19-June-15	Removed 'Preliminary' marking.
	"Introduction": changed min operating voltage to 1.7V.
	"Features": updated sections: Memories, Low-Power Features, QSPI, e.MMC and DACC. Changed ADC to AFE. Changed minimum voltage to 1.7V.
	Updated Figure 2-1, "SAM E70 144-pin Block Diagram".
	Table 2-1 "Configuration Summary": deleted TFBGA64 package option.
	Table 4-1 "Signal Description List": added signals GNDPLL, GNDPLLUSB, GNDANA, GNDUTMI. Changed minimum voltage for VDDIO, VDDIN, VDDPLLUSB and VDDUTMII.
	Updated Table 5-1 "144-lead Package Pinout", Table 5-2 "100-lead Package Pinout" and Table 5-3 "64-lead LQFP Package Pinout".
	Section 6. "Power Considerations" Section 6.2.1 "Power-up": changed min operating voltage to 1.7V for VDDIO and VDDIN. Added constraint regarding overcurrent. Table 6-1 "Power Supplies": changed minimum voltage for VDDIO, VDDIN, VDDPLLUSB and VDDUTMII. Updated column "Associated Ground". Section 6.2.2 "Power-down": added constraint regarding overcurrent. Updated Table 6-2 "Low-power Mode Configuration Summary".
	Section 7. "Input/Output Lines" Removed redundant Section 6.3. TST Pin (already in Section 15. "Debug and Test Features"). Updated Section 7.4 "ERASE Pin".
	Section 10. "Memories" Updated Section 10.1.2 "Tightly Coupled Memory (TCM) Interface" and Section 10.1.4 "Backup SRAM". Updated Section 10.1.5.6 "Unique Identifier".
	Updated Table 13-1 "Peripheral Identifiers".
	Section 14. "ARM Cortex-M7 Processor" Section 14-3 "ARM Cortex-M7 Configuration": changed number of IRQ priority levels.
	Section 15. "Debug and Test Features" Removed redundant Section 14.7.2. NRST Pin and Section 14.7.3. ERASE Pin (already in Section 7. "Input/Output Lines").
	Section 17. "Bus Matrix (MATRIX)" Table 17-4 "Register Mapping": corrected reset values for MATRIX_PRASx and MATRIX_PRBSx registers.
	Section 21. "Enhanced Embedded Flash Controller (EEFC)" Updated Section 21.2 "Embedded Characteristics". Added Figure 21-1, "Flash Memory Areas". Section 21.4.3.7 "Security Bit Protection": added detail on ETM.
	Section 22. "Supply Controller (SUPC)" Figure 22-2, "Separate Backup Supply Powering Scheme": updated figure and corrected min voltage in note on ADC/DAC/ACC.
	Section 22. "Watchdog Timer (WDT)" Section 22.5.1 "Watchdog Timer Control Register": added note on modification of WDT_CR values.. Section 22.5.2 "Watchdog Timer Mode Register": updated note on modification of WDT_MR values.

Downloaded from Arrow.com.

Table 62-2. SAM E70 Datasheet Rev. 11296C – Revision History (Continued)

Date	Changes
19-June-15	Section 25. “Reset Controller (RSTC)” Table 25.5 “Reset Controller (RSTC) User Interface”: updated reset value for RSTC_MR.
	Section 25. “Real-time Clock (RTC)” Added Section 25.6.13 “RTC Write Protection Mode Register”. Updated Table 25-2 “Register Mapping”with RTC_WPMR at offset 0xE4. Added write protection for Section 25.6.1 “RTC Control Register”, Section 25.6.2 “RTC Mode Register”, Section 25.6.5 “RTC Time Alarm Register” and Section 25.6.6 “RTC Calendar Alarm Register”.
	Section 29. “General Purpose Backup Registers (GPBR)” Corrected total size of backup registers.
	Section 25. “Real-time Clock (RTC)” Updated Section 25.5.7 “RTC Accurate Clock Calibration”.
	Section 28. “Clock Generator” Updated Section 28.5.6 “Main Clock Frequency Counter”. Updated Section 28.5.7 “Switching Main Clock between the RC Oscillator and Crystal Oscillator”. Updated Section 28.6.1 “Divider and Phase Lock Loop Programming” with paragraph on correct programming of the multiplication factor of the PLL.
	Section 29. “Power Management Controller (PMC)” Section 29.2 “Embedded Characteristics”: updated bullet on Peripheral Clocks. Updated Figure 29-1, "General Clock Block Diagram". Section 29.8 “Peripheral Clock Controller”: deleted paragraphs about dividing peripheral clocks. Updated Section 29.12 “Core and Bus Independent Clocks for Peripherals”. Updated Section 29.15 “Main Clock Failure Detection”. Section 29.19 “Register Write Protection”: added PMC Clock Generator Main Clock Frequency Register to list of write-protected registers. Updated Table 29-4 “Register Mapping” with PMC_PMMR at offset 0x0130. Section 29.20.9 “PMC Clock Generator Main Clock Frequency Register”: updated MAINF bit description. Section 29.20.17 “PMC Interrupt Mask Register”: added missing bits PCKRDY3–PCKRDY6 (bits 11 to 14). Section 29.20.26 “PMC Peripheral Control Register”: updated PID field description. Deleted DIV field from register table; bits 16 and 17 now reserved. Deleted DIV description. Added Section 29.20.35 “PLL Maximum Multiplier Value Register”.
	Section 32. “Parallel Input/Output Controller (PIO)” Deleted section “Keypad Controller” and all related registers.
	Section 33. “External Bus Interface (EBI)” Section 33.5.3.4 “NAND Flash Support”: changed NCS3 address space.
	Section 35. “DMA Controller (XDMAC)” Section 35.2 “Embedded Characteristics”: added FIFO size. Updated Figure 35-1, "DMA Controller (XDMAC) Block Diagram". Section 35.5.4.1 “Single Block With Single Microblock Transfer”: in Step 6, deleted sub-step to activate a secure channel. Section 35.9.28 “XDMAC Channel x [x = 0..23] Configuration Register”: bit 5 now reserved. Deleted PROT bit description. Updated PERIF field description.
	Section 35. “USB High-Speed Interface (USBHS)” Added Section 35.4.1 “I/O Lines”. Updated Figure 35-2, "General States". Updated Section 35.5.3.3 “Device Detection”. Section 35.6.1 “General Control Register”: added bit 8, VBUSHWC. Section 35.6.2 “General Status Register”, Section 35.6.3 “General Status Clear Register”, Section 35.6.4 “General Status Set Register”: added bit 9, VBUSRQ, VBUSRQC and VBUSRQS and bit descriptions in these registers.

Downloaded from Arrow.com.

Table 62-2. SAM E70 Datasheet Rev. 11296C – Revision History (Continued)

Date	Changes
19-Jun-15	Section 36. “Ethernet MAC (GMAC)” Updated register index range for: - Section 36.8.106 “GMAC Interrupt Status Register Priority Queue x” - Section 36.8.107 “GMAC Transmit Buffer Queue Base Address Register Priority Queue x” - Section 36.8.108 “GMAC Receive Buffer Queue Base Address Register Priority Queue x” - Section 36.8.109 “GMAC Receive Buffer Size Register Priority Queue x” - Section 36.8.115 “GMAC Interrupt Enable Register Priority Queue x” - Section 36.8.116 “GMAC Interrupt Disable Register Priority Queue x” - Section 36.8.117 “GMAC Interrupt Mask Register Priority Queue x” Section 36.8.107 “GMAC Transmit Buffer Queue Base Address Register Priority Queue x”: expanded size of field TXBQBA to 30 bits Section 36.8.108 “GMAC Receive Buffer Queue Base Address Register Priority Queue x”: expanded size of field RXBQBA to 30 bits Section 36.8.117 “GMAC Interrupt Mask Register Priority Queue x”: inverted bit value definitions (‘0’ means enabled, ‘1’ means disabled
	Section 39. “High Speed Multimedia Card Interface (HSMCI)” Updated Table 39-4 “Bus Topology“ (4-bit instead of 8-bit).
	Section 39. “Quad SPI Interface (QSPI)” Section 39.7.2 “QSPI Mode Register”: updated equations and NBBITS description. Section 39.7.9 “QSPI Serial Clock Register”: updated equations. Section 39.7.12 “QSPI Instruction Frame Register”: updated INSTEN bit description.
	Section 41. “Two-wire Interface (TWIHS)” Section 41.6.3.3 “Master Transmitter Mode” and “Read Sequence”: added sentence on clearing TXRDY flag.
	Section 43. “Universal Synchronous Asynchronous Receiver Transmitter (USART)” Section 43.1 “Description” and Section 43.2 “Embedded Characteristics”: updated with content on Modem feature Table 43-1 “I/O Line Description“ added lines RI, DSR, DCD, and DTR. Section 43.5.1 “I/O Lines”: updated section and deleted paragraph on TXD line. Updated Figure 43-2, "Baud Rate Generator". “Baud Rate Calculation Example”, corrected formula. Updated Figure 43-3, "Fractional Baud Rate Generator". Section 43.6.1.3 “Baud Rate in Synchronous Mode or SPI Mode”, corrected formula. Added Section 43.6.7 “Modem Mode”. Section 43.7.1 “USART Control Register”: added bits DTREN and DTRDIS Section 43.7.3 “USART Mode Register” defined USART_MODE field value 0x3 as “MODEM” Section 43.7.5 “USART Interrupt Enable Register”, Section 43.7.9 “USART Interrupt Disable Register”, Section 43.7.13 “USART Interrupt Mask Register”: added bits RIIC, DSRIC, and DCDIC. Section 43.7.6 “USART Interrupt Enable Register (SPI_MODE)”, Section 43.7.10 “USART Interrupt Disable Register (SPI_MODE)”, Section 43.7.14 “USART Interrupt Mask Register (SPI_MODE)”: added bit NSSE. Section 43.7.17 “USART Channel Status Register”: added bits RIIC, DSRIC, DCDIC, DSR, and DCD. Section 43.7.18 “USART Channel Status Register (SPI_MODE)”: added bits NSSE and NSS. Updated Figure 43-42, "Slave Node Synchronization".
	Section 45. “Controller Area Network (MCAN)” Replaced ‘HCLK’ and ‘m_can_hclk’ by ‘peripheral clock’. Replaced ’can_clk’ by ‘CAN core clock’. Replaced ‘tcan_clk’ by ‘tcore clock’. Section 45.5.7 “Message RAM”: deleted sentence on storage constraints. Section 45.4.2 “Power Management”: added recommendations on clock frequencies.
	Section 46. “Timer Counter (TC)” Added important note in Section 46.7.6 “TC Counter Value Register”, Section 46.7.7 “TC Register A”, Section 46.7.8 “TC Register B” and Section 46.7.9 “TC Register C”. Section 46.7.14 “TC Extended Mode Register”: updated TRIGSRCB bit description.

Downloaded from Arrow.com.

Table 62-2. SAM E70 Datasheet Rev. 11296C – Revision History (Continued)

Date	Changes
19-June-15	Section 47. “Pulse Width Modulation Controller (PWM)” Number of fault inputs corrected to 8. Size of dead-time counter/generator corrected to 12 bits. Number of event lines corrected to 2. Number of comparison units corrected to 8. Section 47.7.49 “PWM External Trigger Register” and Section 47.7.50 “PWM Leading-Edge Blanking Register”: corrected register index to 2. Section 47.7.50 “PWM Leading-Edge Blanking Register”: corrected formula for LEBDELAY.
	Section 48. “Analog Front-End Controller (AFEC)” Section 48.6.11 “Input Gain and Offset” changed AOFF configuration value. Corrected formula for offset values. Section 48.7.13 “AFEC Interrupt Status Register”: defined EOCAL bit as ‘cleared on read’. Section 48.7.20 “AFEC Channel Offset Compensation Register”: AOFF field modified to 10 bits (was 12 bits). Bits 10 and 11 now reserved.
	Section 51. “Digital-to-Analog Converter Controller (DACC)” Added information on Bypass mode in: - Section 51.1 “Description” - Section 51.6.4.4 “Bypass Mode” Updated Section 51.6.1 “Digital-to-Analog Conversion”. Added Figure 51-2, "Conversion Sequence in Trigger Mode"and Figure 51-3, "Conversion Sequence in Free-running Mode". Updated Section 51.6.4.3 “Max Speed Mode” and added Figure 51-4, "Conversion Sequence in Max Speed Mode". Deleted Section 7.6 “DACC Timings”. Section 51.7.2 “DACC Mode Register”: added bit ZERO (bit 5) and bit description. Removed bits ENDTX0, ENDTX1, TXBUFE0 and TXBUFE1 from Section 51.7.8 “DACC Interrupt Enable Register”, Section 51.7.9 “DACC Interrupt Disable Register”, Section 51.7.10 “DACC Interrupt Mask Register” and Section 51.7.11 “DACC Interrupt Status Register”.
	Section 56. “Electrical Characteristics” All content (tables and text) updated from Section 56.2 to Section 56.12. “Embedded Flash Characteristics” is now Section 56.12 (moved from sections Section 56.13 and Section 56.14) Section 56.14.1.7 “SDRAM Timings”: modfied content. Updated Table 56-64 “USART SPI Timings” and Table 56-86 “USART SPI Timings”.
	Section 57. “Mechanical Characteristics” Updated Figure 57-4, "100-pin LQFP Package Mechanical Drawing" and Figure 57-6, "64-pin LQFP Package Mechanical Drawing".
	Added Section 58. “Schematic Checklist”.
	Added Section 61. “Errata”.

Downloaded from Arrow.com.

Table 62-3. SAM E70 Datasheet Rev. 11296B – Revision History

Date	Changes
24-Feb-15	"Description": updated details on PWM, 16-bit timers, RTC, RTT and Backup mode.
	"Features": updated details on PWM.
	Section 1. "Configuration Summary" Table 1-1 "Configuration Summary": Modifications made to Timer Counter Channels I/O, USART/UART, QSPI, SPI, USART SPI.
	Section 2. "Block Diagram": added AHBP block. Added Backup RAM block. Removed TRACECTL. Changed block name to Serial Wire Debug/JTAG Boundary Scan (was JTAG and Serial Wire). Modified signal names to VREFP and VREFN (were ADVREFP and ADVREFN).
	Section 3. "Signal Description" Table 3-1 "Signal Description List": corrected upper index for Two-wire Interface - TWIHS. Modified signal names to VREFP and VREFN (were ADVREFP and ADVREFN). In section FFPI, corrected upper index of signal PGMEN to '1' and removed signal PGMCK.
	Section 5. "Package and Pinout" In all pinout tables, modified signal names to VREFP and VREFN (were ADVREFP and ADVREFN). Replaced tables "Pinout for 144-pin LQFP Package" and "Pinout for 144-pin LFBGA Package" with single Table 5-1 "144-lead Package Pinout" and reworked the table. For Pin 110/PIOD: replaced TRACECTL with '–'. Added notes to all signals in column 'Alternate' for details on selecting extra functions and system functions. Replaced tables "Pinout for 100-pin LQFP Package" and "Pinout for 100-ball TFBA Package"by single Table 5-2 "100-lead Package Pinout" and reworked the table. Reworked table "Pinout for 64-pin LQFP Package" and renamed it to Table 5-3 "64-lead Package Pinout".
	Section 6. "Power Considerations" Section 6.2 "Power Constraints": updated constraint for VDDCORE, VDDPLL and VDDUTMIC. Section 6.2.1 "Power-up": changed value of rising slope of VDDIO and VDDIN to 2.4V/ms. Section 6.2.2 "Power-down": added detail on VDDCORE falling slope.
	Section 7. "Input/Output Lines" Section 7.1 "General-Purpose I/O Lines": changed ODT to R_{SERIAL} in text and figure. Section 7.2.2 "Embedded Trace Module (ETM) Pins"; removed TRACECTL Section 7.5 "ERASE Pin": added details on in-situ reprogrammability.
	Section 10. "Memories" Table 10-1 "TCM Configurations in Kbytes": corrected column GPNVM Bit [8:7] by inverting values (0 first, 3 last). Table 10-4 "General-purpose Non volatile Memory Bits": GPNVM bit 1: inverted 0 and 1 values. GPNVM bit 7–8: inverted all values for TCM configuration and added Note. Section 10.1.1 "Internal SRAM": updated section. Section 10.1.2 "Tightly Coupled Memory (TCM) Interface": added detail on enable/disable of ITCM/DTCM. Section 10.1.4 "Backup SRAM": updated SRAM address. Removed detail on read/write accesses. Section 10.1.5 "Flash Memories": added details on the attribute definitions for programming operations vs. fetch/read operations. Section 10.1.5.9 "Fast Flash Programming Interface": removed 'serial JTAG interface'.
	Section 11. "Event System" Table 11-1 " Event Mapping List": in row "Audio clock recovery from Ethernet' changed the text in Description column.
	Section 13. "Peripherals" Table 13-1 "Peripheral Identifiers": modfied content of column 'Description' for clarity. Section 13.2 "Peripheral Signal Multiplexing on I/O Lines": corrected PIOC to PIOD for 100-pin version. Moved Section 13.3 "Peripheral Mapping to DMA" to Section 35.3 "DMA Controller Peripheral Connections".

Downloaded from Arrow.com.

Table 62-3. SAM E70 Datasheet Rev. 11296B – Revision History (Continued)

Date	Changes
24-Feb-15	Section 15. “Debug and Test Features” Section 15.1 “Description”: removed references to JTAG Debug Port and JTAG-DP. Updated Figure 15-1 “Debug and Test Block Diagram”: added Cortex-M7, ETM and PCK3 blocks and trace pins. Renamed block ‘SWJ-DP’ to ‘SW-DP’. Table 15-1 “Debug and Test Signal List”: removed TRACECTL. Updated Figure 15-4 “Debug Architecture”. added ETM and Trace Port blocks. Removed TPIU. Section 15.6.5 “Serial Wire Debug Port (SW-DP) Pins”: removed all references to JTAG Debug Port and JTAG-DP. Section 15.6.6 “Embedded Trace Module (ETM) Pins”: removed TRACECTL from bullet points. Updated Section 15.6.7 “Flash Patch Breakpoint (FPB)” . Section 15.6.9.2 “Asynchronous Mode”: removed reference to JTAG Debug Port and JTAG debug mode.
	Section 16. “SAM-BA Boot Program” Section 16.6.4 “In Application Programming (IAP) Feature”: replaced software code example.
	Section 18. “Bus Matrix (MATRIX)” Table 18-3 “Master to Slave Access”: changed Master 4/Slave 4 access from possible (“x”) to not possble (‘-”) Table 18-4 “Register Mapping”: changed reset value for CCFG_SYSIO register. Section 18.12.7 “System I/O and CAN1 Configuration Register”: corrected typo in CAN1DMABA bit name. Section 18.11 “Register Write Protection”: replaced “The WPVS bit is automatically cleared after reading the MATRIX_WPSR” with “The WPVS flag is reset by writing the MATRIX_WPMR with the appropriate access key WPKEY” Section 18.12.10 “Write Protection Status Register”: in WPVS bit description, replaced two instances of “since the last read of the MATRIX_WPSR” with “since the last write of the MATRIX_WPMR”.
	Section 21. “Enhanced Embedded Flash Controller (EEFC)” Section 21.4.3.2 “Write Commands”: added information on DMA write accesses.
	Section 30. “Power Management Controller (PMC)” Section 30.9 “Asynchronous Partial Wake-up”: inserted new sub-section “Asynchronous Partial Wake-up in Wait Mode (SleepWalking)” to better describe SleepWalking. Section 30.10 “Free-Running Processor Clock”: removed reference to MCK.
	Section 31. “Parallel Input/Output Controller (PIO)” Section 31.2 “Embedded Characteristics”: added bullet on Programmable I/O Drive. Added Section 31.5.12 “Programmable I/O Drive”. Section 31.5.15.4 “Programming Sequence”: “With DMA”: in fifth step, replaced reference to BTCx with ‘DMA status flag to indicate that the buffer transfer is complete’ Table 31-5 “Register Mapping”: added PIO_DRIVER register at offset 0x0118 and added Section 31.6.49 “PIO I/O Drive Register”.
	Section 35. “DMA Controller (XDMAC)” Added Section 35.3 “DMA Controller Peripheral Connections”.
	Section 37. “USB High-Speed Interface (USBHS)” Table 37-1 “Description of USB Pipes/Endpoints“; corrected data in columns ‘DMA’ and ‘High Bandwidth’. Modified signal names to HSDM/DM and HSDP/DP in Figure 37-1 “USBHS Block Diagram” and Table 37-2 “Signal Description”. Updated descriptions. Removed Section 37.3.1 “Application Block Diagram” and Figures 37-2, 37-3 and 37-4. Removed Section 37.4.1 “I/O Lines”. Modified Section 37.5.3.3 “Device Detection”. Section 37.6.2 “General Status Register”, Section 37.6.3 “General Status Clear Register”, Section 37.6.4 “General Status Set Register”: removed bit VBUSRQ and bit description. Bit 9 now reserved in these registers.

Downloaded from Arrow.com.

Table 62-3. SAM E70 Datasheet Rev. 11296B – Revision History (Continued)

Date	Changes
24-Feb-15	Section 38. “Ethernet MAC (GMAC)” Section 38.8.13 “GMAC Interrupt Mask Register”: corrected general bit description (swapped definitions provided for 0: and 1:)
	Section 40. “Quad SPI Interface (QSPI)” Section 40.5.4 “Direct Memory Access Controller (DMA)”: added Note on 32-bit aligned DMA write accesses. Figure 40-9 “Instruction Transmission Flow Diagram”: modified text if TFRTYP = 0 Section 40.6.7 “Register Write Protection”: added Scrambling Mode Register and Scrambling Key Register to the list of registers that can be write-protected. Section 40.7.13 “QSPI Scrambling Mode Register” and Section 40.7.14 “QSPI Scrambling Key Register”: added "This register can only be written if bit WPEN is cleared in the QSPI Write Protection Mode Register.".
	Section 42. “Two-wire Interface (TWIHS)” Replaced all instances of ‘BTC’ with ‘DMA status flag’.
	Section 47. “Controller Area Network (MCAN)” Figure 47-1 “MCAN Block Diagram”: added Note. Section 47.4.2 “Power Management”: added recommendations for CAN clock frequency. Added Section 47.4.4 “Address Configuration”.
	Section 48. “Timer Counter (TC)” Replaced occurrences of ‘quadrature decoder logic’with ‘quadrature decoder’ or ‘QDEC’ throughout the document. Section 48.7.14 “TC Extended Mode Register”: changed description for field TRIGSRCB for value 1.
	Section 49. “Pulse Width Modulation Controller (PWM)” Section 49.5.3 “Interrupt Sources”: removed the following sentence: “Note that it is not recommended to use the PWM interrupt line in Edge-sensitive mode.” “Method 3: Automatic write of duty-cycle values and automatic trigger of the update”: removed reference to non-existant field BTC. Modified Figure 49-28 “External PWM Reset Mode: Power Factor Correction Application”. Removed R_{LIMIT} and Zener diode from Figure 49-32 “Cycle-By-Cycle Duty Mode: LED String Control”.
	Section 50. “Analog Front-End Controller (AFEC)” In text and tables throughout this section, all occurrences of ADVREF have been modified to VREFP. Figure 50-1, "Analog Front-End Controller Block Diagram": added 2nd DAC. Removed ADVREF; added VREFP and VREFN. Table 50-1 “AFEC Signal Description”: removed row with VDDANA. Added row with VREFN. Section 50.5 “Product Dependencies”: reorganized sub-sections. In Section 50.5.2 “Power Management”, added sentence on Sleep mode. Modified Section 50.5.1 “I/O Lines”. Removed section 50.5.3 Analog Inputs. Section 50.6.1 “Analog Front-End Conversion”: changed PRESCAL condition from ‘0’ to ‘1’ for frequency range fperipheral clock/2. Figure 50-7 “Analog Full Scale Ranges in Single-Ended/Differential Applications Versus Gain”: replaced all occurrences of VADVREF with VVREFP; replaced min ‘0’ value with VVREFN=0. Section 50.7.2 “AFEC Mode Register”: modified PRESCAL description.

Downloaded from Arrow.com.

Table 62-3. SAM E70 Datasheet Rev. 11296B – Revision History (Continued)

Date	Changes
24-Feb-15	Section 51. “Digital-to-Analog Converter (DACC)” Section 51.1 “Description”: removed information on refresh feature. Figure 51-1 “Block Diagram”: added VDDANA, VREFP and VREFN. Table 51-1 “DACC Signal Description”: added VREFP and VREFN to table. Section 51.2 “Embedded Characteristics”: removed bullet on refresh period. Added Section 51.5.1 “I/O Lines”. Section 51.6.3 “Analog Output Mode Selection”: corrected bit name for output modeselection to ‘DIFF’ from ‘ANA_MODE_SEL’ . Section 51.6.4 “Conversion Modes”: added details on enabling conversion modes. Removed bullet “Interpolated Mode”. Removed section 51.6.5 “Refresh Mode”. Updated Section 51.6.4.4 “Interpolation Mode”. Section 51.7.2 “DACC Mode Register”: removed field REFRESH and description. Bits 15:8 now reserved. Section 51.7.6 “DACC Channel Status Register”: modified DACRDYx bit descriptions. Section 51.7.11 “DACC Interrupt Status Register”: ENDTXx, TXBUFEx descriptions: corrected register name to ‘DACC_CDRx’ from ‘DACC _TCR or DACC_TNCR’.
	Section 52. “Analog Comparator Controller (ACC)” In text and in tables throughout this section, changed all occurrences of ADVREF to VREFP. Section 52.2 “Embedded Characteristics”: In bullet: “Four Voltage References...”, changed ADVREF to ‘External Voltage Reference’ Renamed Section 5. to “Signal Description” Removed Table 52-1 “List of External Analog Data Inputs” and note referring to this table.
	Section 53. “Integrity Check Monitor (ICM)” Section 53.1 “Description”: updated content. Renamed section “ICM SHA Engine” to “Using ICM as SHA Engine” and updated content. Added Section 53.5.4.1 “Settings for Simple SHA Calculation”. Section 53.5.2.2 “ICM Region Configuration Structure Member”: updated descriptions for RHIEN, DMIEN, BEIEN, WCIEN, ECIEN, SUIEN and MPROT. Section 53.6.1 “ICM Configuration Register”: updated descriptions for DAPROT and HAPROT. Section 53.6.3 “ICM Status Register”: updated descriptions for RAWRMDIS and RMDIS.

Downloaded from Arrow.com.

Table 62-3. SAM E70 Datasheet Rev. 11296B – Revision History (Continued)

Date	Changes
	Section 55. “Advanced Encryption Standard (AES)” Section 55.4.5.2 “DMA Mode”: removed references to ‘BTC’ throughout.
	Section 56. “Electrical Characteristics” Table 56-2 “DC Characteristics”: updated conditions for V_{IL}, V_{IH}, V_{OH}, V_{OL}, I_O, R_{PULLUP}, $R_{PULLDOWN}$, $R_{SERIAL.}$ Added parameter Flash Active Current characteristics. Added parameter Static Current. Modified Note (1) below table. Table 56-3 “1.2V Voltage Regulator Characteristics”: removed note on VDDIO voltage at power-up (was Note 3). Updated note on VDDIO voltage value. Changed values of CD_{OUT}. Changed conditions for parameter t_{START} and CD_{OUT} value in Note 2. Table 56-4 “Core Power Supply Brownout Detector Characteristics”: updated all values. Changed Note 1. Table 56-6 “VDDIO Supply Monitor”: updated values for $T_{ACCURACY}$ Table 56-9. “DC Flash Characteristics” moved to Table 56-2 “DC Characteristics”. Section 56.3.2.1 “Sleep Mode Conditions”: corrected number of WKUP pins. Added Section 56.3.6 “I/O Switching Power Consumption”. Table 56-21 “32 kHz RC Oscillator Characteristics”: changed max values to TBD for t_{START}, I_{DDON} and $I_{DDON_STANDBY}$. Table 56-25 “3 to 20 MHz Crystal Oscillator Characteristics”: for t_{START} and $I_{DD_ON,}$ changed max values to TBD. Added parameter $I_{DD._STANDBY.}$ Table 56-26 “Crystal Characteristics”: ESR: added new row with condition Fundamental at 3 MHz. Changed max values for 8 and 12 MHz. Table 56-29 “PLLA Characteristics”: changed max value of f_{IN}. Added parameter I_{DD_STDBY} Added Section 56.6 “PLLUSB Characteristics”> Updated section Section 56.7 “USB Transceiver Characteristics”. Section 56.9 “AFE Characteristics”: changed numbering of sub-sections throughout.
24-Feb-15	- Removed bullet on min and max data. - Changed all occurrences of ADVREFP to VREP, and of ADVREFN to VREFN throughout section. - Changed all occurrences of ADC to AFE, where relevant. - Modified Figure 56-11 “Single-ended Mode AFE” and Figure 56-12 “Differential Mode AFE”. - Table 56-36 “Power Supply Characteristics”: updated I_{VDDIN} conditions in and changed max values. Changed max values for $I_{VDDCORE.}$ Removed Note 1 due to incorrect cross-reference. Added Note 3 on current consumption. - Table 56-38 “VREFP Electrical Characteristics”: changed min and max values for $I_{VREFP.}$ - Table 56-46 “Single-ended Output Offset Error”: added note on voltage application. - Table 56-47 “Single-ended Static Electrical Characteristics”: added conditions and values. - Table 56-49 “Differential Static Electrical Characteristics”: changed min and max values. Added Section 56.10 “Analog Comparator Characteristics”. Section 56.12 “12-bit DAC Characteristics” - Added note to Table 56-59 “Analog Power Supply Characteristics”. Added new conditions to Table 56-62 “Static Performance Characteristics”. and updated min and max values for INL, DNL and Gain Error. Section 56.13 “Timings for Worst-Case Conditions” - Table 56-68 “I/O Characteristics”: new conditions and the corresponding max values added. - Section 56.13.2 “Embedded Flash Characteristics”: in Table 56-87 “AC Flash Characteristics” changed Full Chip Erase values. Replaced two “Embedded Flash Wait State” tables with single Table 56-88 “Embedded Flash Wait State at 105°C“
	Section 56.14 “Timings for STH Conditions” - Table 56-92 “I/O Characteristics”: new conditions and the corresponding max values added. - Section 56.14.2 “Embedded Flash Characteristics”: replaced two “Embedded Flash Wait State” tables with single Table 56-112 “Embedded Flash Wait State at 105°C“

Downloaded from Arrow.com.

Table 62-4. SAM E70 Datasheet Rev. 11296A – Revision History

Date	Changes
15-Oct-13	First issue

Downloaded from Arrow.com.

Table of Contents

Introduction . . . 1

Features . . . 1

1. Description . . . 4

2. Configuration Summary . . . 5

3. Block Diagram . . . 7

4. Signal Description . . . 8

5. Package and Pinout . . . 14

5.1 144-lead Packages . . . 15
5.2 144-lead Package Pinout . . . 16
5.3 100-lead Packages . . . 21
5.4 100-lead Package Pinout . . . 22
5.5 64-lead Package . . . 26
5.6 64-lead Package Pinout . . . 27

6. Power Considerations . . . 30

6.1 Power Supplies . . . 30
6.2 Power Constraints . . . 30
6.3 Voltage Regulator . . . 32
6.4 Backup SRAM Power Switch . . . 32
6.5 Active Mode . . . 32
6.6 Low-power Modes . . . 33
6.7 Wake-up Sources . . . 36
6.8 Fast Startup . . . 36

7. Input/Output Lines . . . 37

7.1 General-Purpose I/O Lines . . . 37
7.2 System I/O Lines . . . 38
7.3 NRST Pin . . . 39
7.4 ERASE Pin . . . 39

8. Interconnect . . . 40

9. Product Mapping . . . 41

10. Memories . . . 42

10.1 Embedded Memories . . . 42
10.2 External Memories . . . 47

11. Event System . . . 48

11.1 Embedded Characteristics . . . 48
11.2 Real-time Event Mapping . . . 49

12. System Controller . . . 52

12.1 System Controller and Peripherals Mapping . . . 52
12.2 Power-on-Reset, Brownout and Supply Monitor . . . 52
12.3 Reset Controller . . . 53

Downloaded from Arrow.com.

13. Peripherals 54
13.1 Peripheral Identifiers 54
13.2 Peripheral Signal Multiplexing on I/O Lines 57

14. ARM Cortex-M7 Processor 58
14.1 Description 58
14.2 Embedded Characteristics 59
14.3 Block Diagram 60
14.4 Programmer's Model 61
14.5 ARM Cortex-M7 Configuration 73

15. Debug and Test Features 74
15.1 Description 74
15.2 Embedded Characteristics 74
15.3 Associated Documents 74
15.4 Debug and Test Block Diagram 75
15.5 Debug and Test Pin Description 75
15.6 Application Examples 76
15.7 Functional Description 78

16. SAM-BA Boot Program 83
16.1 Description 83
16.2 Embedded Characteristics 83
16.3 Hardware and Software Constraints 83
16.4 Flow Diagram 84
16.5 Device Initialization 84
16.6 SAM-BA Monitor 85

17. Fast Flash Programming Interface (FFPI) 89
17.1 Description 89
17.2 **Embedded Characteristics** 89
17.3 Parallel Fast Flash Programming 90

18. Bus Matrix (MATRIX) 98
18.1 Description 98
18.2 **Embedded Characteristics** 98
18.3 Functional Description 101
18.4 Bus Matrix (MATRIX) User Interface 106

19. USB Transmitter Macrocell Interface (UTMI) 118
19.1 Description 118
19.2 Embedded Characteristics 118
19.3 USB Transmitter Macrocell Interface (UTMI) User Interface 119

20. Chip Identifier (CHIPID) 122
20.1 Description 122
20.2 Embedded Characteristics 122
20.3 Chip Identifier (CHIPID) User Interface 124

21. Enhanced Embedded Flash Controller (EEFC) 129
21.1 **Description** 129
21.2 **Embedded Characteristics** 129
21.3 Product Dependencies 129

Downloaded from Arrow.com.

21.4 Functional Description . . . 130
21.5 Enhanced Embedded Flash Controller (EEFC) User Interface . . . 149

22. Supply Controller (SUPC) . . . 157
22.1 Description . . . 157
22.2 Embedded Characteristics . . . 157
22.3 Block Diagram . . . 158
22.4 Functional Description . . . 159
22.5 Supply Controller (SUPC) User Interface . . . 171

23. Watchdog Timer (WDT) . . . 182
23.1 Description . . . 182
23.2 Embedded Characteristics . . . 182
23.3 Block Diagram . . . 183
23.4 Functional Description . . . 184
23.5 Watchdog Timer (WDT) User Interface . . . 186

24. Reinforced Safety Watchdog Timer (RSWDT) . . . 191
24.1 Description . . . 191
24.2 Embedded Characteristics . . . 191
24.3 Block Diagram . . . 192
24.4 Functional Description . . . 193
24.5 Reinforced Safety Watchdog Timer (RSWDT) User Interface . . . 195

25. Reset Controller (RSTC) . . . 199
25.1 Description . . . 199
25.2 Embedded Characteristics . . . 199
25.3 Block Diagram . . . 199
25.4 Functional Description . . . 200
25.5 Reset Controller (RSTC) User Interface . . . 206

26. Real-time Clock (RTC) . . . 210
26.1 Description . . . 210
26.2 Embedded Characteristics . . . 210
26.3 Block Diagram . . . 211
26.4 Product Dependencies . . . 211
26.5 Functional Description . . . 212
26.6 Real-time Clock (RTC) User Interface . . . 219

27. Real-time Timer (RTT) . . . 236
27.1 Description . . . 236
27.2 Embedded Characteristics . . . 236
27.3 Block Diagram . . . 236
27.4 Functional Description . . . 237
27.5 Real-time Timer (RTT) User Interface . . . 239

28. SDRAM Controller (SDRAMC) . . . 244
28.1 Description . . . 244
28.2 Embedded Characteristics . . . 244
28.3 Signal Description . . . 245
28.4 Software Interface/SDRAM Organization, Address Mapping . . . 246
28.5 Product Dependencies . . . 247

Downloaded from Arrow.com.

28.6 Functional Description . . . 251
28.7 SDRAM Controller (SDRAMC) User Interface . . . 258

29. General Purpose Backup Registers (GPBR) . . . 274
29.1 Description . . . 274
29.2 Embedded Characteristics . . . 274
29.3 General Purpose Backup Registers (GPBR) User Interface . . . 275

30. Clock Generator . . . 277
30.1 Description . . . 277
30.2 Embedded Characteristics . . . 277
30.3 Block Diagram . . . 278
30.4 Slow Clock . . . 279
30.5 Main Clock . . . 279
30.6 Divider and PLL Block . . . 283
30.7 UTMI Phase Lock Loop Programming . . . 284

31. Power Management Controller (PMC) . . . 285
31.1 Description . . . 285
31.2 Embedded Characteristics . . . 285
31.3 Block Diagram . . . 286
31.4 Master Clock Controller . . . 286
31.5 Processor Clock Controller . . . 287
31.6 SysTick External Clock . . . 287
31.7 USB Clock Controller . . . 287
31.8 Peripheral Clock Controller . . . 287
31.9 Asynchronous Partial Wake-up . . . 288
31.10 Free-Running Processor Clock . . . 290
31.11 Programmable Clock Output Controller . . . 290
31.12 Core and Bus Independent Clocks for Peripherals . . . 290
31.13 Fast Startup . . . 290
31.14 Startup from Embedded Flash . . . 292
31.15 Main Clock Failure Detection . . . 292
31.16 32.768 kHz Crystal Oscillator Frequency Monitor . . . 294
31.17 Programming Sequence . . . 294
31.18 Clock Switching Details . . . 296
31.19 Register Write Protection . . . 299
31.20 Power Management Controller (PMC) User Interface . . . 301

32. Parallel Input/Output Controller (PIO) . . . 344
32.1 Description . . . 344
32.2 Embedded Characteristics . . . 345
32.3 Block Diagram . . . 346
32.4 Product Dependencies . . . 347
32.5 Functional Description . . . 348
32.6 Parallel Input/Output Controller (PIO) User Interface . . . 362

33. External Bus Interface (EBI) . . . 420
33.1 Description . . . 420
33.2 Embedded Characteristics . . . 420
33.3 EBI Block Diagram . . . 421
33.4 I/O Lines Description . . . 422

Downloaded from Arrow.com.

33.5 Application Example 423

34. Static Memory Controller (SMC) 427
34.1 Description 427
34.2 **Embedded Characteristics** 427
34.3 I/O Lines Description 428
34.4 Multiplexed Signals 428
34.5 Product Dependencies 429
34.6 External Memory Mapping 431
34.7 Connection to External Devices 431
34.8 Application Example 435
34.9 Standard Read and Write Protocols 437
34.10 Scrambling/Unscrambling Function 447
34.11 Automatic Wait States 447
34.12 Data Float Wait States 452
34.13 External Wait 457
34.14 Slow Clock Mode 463
34.15 Asynchronous Page Mode 465
34.16 Static Memory Controller (SMC) User Interface 467

35. DMA Controller (XDMAC) 478
35.1 Description 478
35.2 Embedded Characteristics 478
35.3 Block Diagram 479
35.4 DMA Controller Peripheral Connections 480
35.5 Functional Description 482
35.6 Linked List Descriptor Operation 487
35.7 XDMAC Maintenance Software Operations 489
35.8 XDMAC Software Requirements 490
35.9 Extensible DMA Controller (XDMAC) User Interface 491

36. Image Sensor Interface (ISI) 525
36.1 Description 525
36.2 Embedded Characteristics 526
36.3 Block Diagram 526
36.4 Product Dependencies 527
36.5 Functional Description 528
36.6 Image Sensor Interface (ISI) User Interface 538

37. USB High-Speed Interface (USBHS) 570
37.1 Description 570
37.2 Embedded Characteristics 570
37.3 Block Diagram 571
37.4 Product Dependencies 572
37.5 Functional Description 573
37.6 USB High-Speed (USBHS) User Interface 599

38. Ethernet MAC (GMAC) 705
38.1 Description 705
38.2 Embedded Characteristics 705
38.3 Block Diagram 706
38.4 Signal Interface 706

Downloaded from Arrow.com.

38.5 Product Dependencies . . . 707
38.6 Functional Description . . . 708
38.7 Programming Interface . . . 735
38.8 Ethernet MAC (GMAC) User Interface . . . 739

39. High Speed Multimedia Card Interface (HSMCI) . . . 873
39.1 Description . . . 873
39.2 Embedded Characteristics . . . 873
39.3 Block Diagram . . . 874
39.4 Application Block Diagram . . . 875
39.5 Pin Name List . . . 875
39.6 Product Dependencies . . . 876
39.7 Bus Topology . . . 876
39.8 High Speed MultiMedia Card Operations . . . 879
39.9 SD/SDIO Card Operation . . . 888
39.10 CE-ATA Operation . . . 889
39.11 HSMCI Boot Operation Mode . . . 890
39.12 HSMCI Transfer Done Timings . . . 891
39.13 Register Write Protection . . . 892
39.14 High Speed MultiMedia Card Interface (HSMCI) User Interface . . . 893

40. Serial Peripheral Interface (SPI) . . . 922
40.1 Description . . . 922
40.2 Embedded Characteristics . . . 922
40.3 Block Diagram . . . 923
40.4 Application Block Diagram . . . 923
40.5 Signal Description . . . 924
40.6 Product Dependencies . . . 924
40.7 Functional Description . . . 925
40.8 Serial Peripheral Interface (SPI) User Interface . . . 938

41. Quad SPI Interface (QSPI) . . . 955
41.1 Description . . . 955
41.2 Embedded Characteristics . . . 955
41.3 Block Diagram . . . 956
41.4 Signal Description . . . 956
41.5 Product Dependencies . . . 957
41.6 Functional Description . . . 958
41.7 Quad SPI Interface (QSPI) User Interface . . . 977

42. Two-wire Interface (TWIHS) . . . 997
42.1 Description . . . 997
42.2 Embedded Characteristics . . . 997
42.3 List of Abbreviations . . . 998
42.4 Block Diagram . . . 998
42.5 Product Dependencies . . . 999
42.6 Functional Description . . . 1000
42.7 Two-wire Interface High Speed (TWIHS) User Interface . . . 1040

43. Synchronous Serial Controller (SSC) . . . 1063
43.1 Description . . . 1063
43.2 **Embedded Characteristics** . . . 1063

Downloaded from Arrow.com.

43.3 Block Diagram . . . 1064
43.4 Application Block Diagram . . . 1064
43.5 SSC Application Examples . . . 1065
43.6 Pin Name List . . . 1066
43.7 Product Dependencies . . . 1067
43.8 Functional Description . . . 1068
43.9 Synchronous Serial Controller (SSC) User Interface . . . 1081

44. Inter-IC Sound Controller (I2SC) . . . 1108
44.1 Description . . . 1108
44.2 Embedded Characteristics . . . 1108
44.3 Block Diagram . . . 1109
44.4 I/O Lines Description . . . 1109
44.5 Product Dependencies . . . 1110
44.6 Functional Description . . . 1111
44.7 I2SC Application Examples . . . 1115
44.8 Inter-IC Sound Controller (I2SC) User Interface . . . 1117

45. Universal Synchronous Asynchronous Receiver Transceiver (USART) . . . 1130
45.1 Description . . . 1130
45.2 Embedded Characteristics . . . 1130
45.3 Block Diagram . . . 1132
45.4 I/O Lines Description . . . 1132
45.5 Product Dependencies . . . 1133
45.6 Functional Description . . . 1134
45.7 Universal Synchronous Asynchronous Receiver Transmitter (USART) User Interface . . . 1190

46. Universal Asynchronous Receiver Transmitter (UART) . . . 1259
46.1 Description . . . 1259
46.2 Embedded Characteristics . . . 1259
46.3 Block Diagram . . . 1259
46.4 Product Dependencies . . . 1260
46.5 Functional Description . . . 1261
46.6 Universal Asynchronous Receiver Transmitter (UART) User Interface . . . 1271

47. Controller Area Network (MCAN) . . . 1284
47.1 Description . . . 1284
47.2 Embedded Characteristics . . . 1284
47.3 Block Diagram . . . 1285
47.4 Product Dependencies . . . 1286
47.5 Functional Description . . . 1287
47.6 Controller Area Network (MCAN) User Interface . . . 1317

48. Timer Counter (TC) . . . 1373
48.1 Description . . . 1373
48.2 Embedded Characteristics . . . 1373
48.3 Block Diagram . . . 1374
48.4 Pin List . . . 1375
48.5 Product Dependencies . . . 1376
48.6 Functional Description . . . 1378
48.7 Timer Counter (TC) User Interface . . . 1400

Downloaded from Arrow.com.

49. Pulse Width Modulation Controller (PWM) 1433
49.1 Description 1433
49.2 Embedded Characteristics 1434
49.3 Block Diagram 1435
49.4 I/O Lines Description 1436
49.5 Product Dependencies 1436
49.6 Functional Description 1440
49.7 Pulse Width Modulation Controller (PWM) User Interface 1481

50. Analog Front-End Controller (AFEC) 1537
50.1 Description 1537
50.2 Embedded Characteristics 1538
50.3 Block Diagram 1539
50.4 Signal Description 1539
50.5 Product Dependencies 1540
50.6 Functional Description 1542
50.7 Analog Front-End Controller (AFEC) User Interface 1561

51. Digital-to-Analog Converter Controller (DACC) 1594
51.1 Description 1594
51.2 Embedded Characteristics 1594
51.3 Block Diagram 1595
51.4 Signal Description 1595
51.5 Product Dependencies 1596
51.6 Functional Description 1597
51.7 Digital-to-Analog Converter Controller (DACC) User Interface 1604

52. Analog Comparator Controller (ACC) 1619
52.1 Description 1619
52.2 Embedded Characteristics 1619
52.3 Block Diagram 1620
52.4 Signal Description 1620
52.5 Product Dependencies 1621
52.6 Functional Description 1622
52.7 Analog Comparator Controller (ACC) User Interface 1623

53. Integrity Check Monitor (ICM) 1633
53.1 Description 1633
53.2 Embedded Characteristics 1634
53.3 Block Diagram 1634
53.4 Product Dependencies 1635
53.5 Functional Description 1636
53.6 Integrity Check Monitor (ICM) User Interface 1649

54. True Random Number Generator (TRNG) 1663
54.1 Description 1663
54.2 Embedded Characteristics 1663
54.3 Block Diagram 1663
54.4 Product Dependencies 1663
54.5 Functional Description 1664
54.6 True Random Number Generator (TRNG) User Interface 1665

Downloaded from Arrow.com.

55. Advanced Encryption Standard (AES) 1672
55.1 Description 1672
55.2 Embedded Characteristics 1672
55.3 Product Dependencies 1673
55.4 Functional Description 1674
55.5 Advanced Encryption Standard (AES) User Interface 1686

56. Electrical Characteristics 1705
56.1 Absolute Maximum Ratings 1705
56.2 DC Characteristics 1706
56.3 Power Consumption 1712
56.4 Oscillator Characteristics 1721
56.5 PLLA Characteristics 1729
56.6 PLLUSB Characteristics 1729
56.7 USB Transceiver Characteristics 1730
56.8 AFE Characteristics 1731
56.9 Analog Comparator Characteristics 1745
56.10 Temperature Sensor 1746
56.11 12-bit DAC Characteristics 1747
56.12 Embedded Flash Characteristics 1751
56.13 Timings for Worst-Case Conditions 1753
56.14 Timings for STH Conditions 1780

57. Mechanical Characteristics 1807
57.1 144-pin LQFP Package 1807
57.2 144-ball LFBGA Package 1808
57.3 144-ball UFBGA Package 1809
57.4 100-pin LQFP Package 1810
57.5 100-ball TFBGA Package 1811
57.6 64-pin LQFP Package 1812
57.7 Soldering Profile 1813
57.8 Packaging Resources 1813

58. Schematic Checklist 1814
58.1 Power Supplies 1814
58.2 General Hardware Recommendations 1820
58.3 Boot Program Hardware Constraints 1835

59. Marking 1836

60. Ordering Information 1837

61. Errata 1838
61.1 AFE Controller (AFEC) 1838
61.2 AHB Peripheral Port (AHBP) 1838
61.3 AHB Slave Port (AHBS) 1838
61.4 ARM Cortex-M7 1839
61.5 Extended DMA Controller (XDMAC) 1839
61.6 Fast Flash Programming Interface (FFPI) 1839
61.7 Inter-IC Sound Controller (I2SC) 1840
61.8 Master CAN-FD Controller (MCAN) 1840
61.9 Power Management Controller (PMC) 1840

Downloaded from Arrow.com.

61.10 Quad SPI Interface (QSPI) 1841
61.11 Serial Synchronous Controller (SSC) 1841
61.12 Supply Controller (SUPC) 1841
61.13 TWI High Speed (TWIHS) 1842
61.14 Universal Synchronous Asynchronous Receiver Transmitter (USART) 1842
61.15 USB High Speed (USBHS) 1842

62. Revision History 1844

Table of Contents 1860

Downloaded from Arrow.com.

Downloaded from Arrow.com.







Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**

© 2016 Atmel Corporation. / Rev.: Atmel-11296D-ATARM-SAM E70-Datasheet_19-Jan-16.

Atmel®, Atmel logo and combinations thereof, Enabling Unlimited Possibilities®, and others are registered trademarks or trademarks of Atmel Corporation in U.S. and other countries. ARM®, ARM Connected® logo, and others are the registered trademarks or trademarks of ARM Ltd. Other terms and product names may be trademarks of others.

DISCLAIMER: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

SAFETY-CRITICAL, MILITARY, AND AUTOMOTIVE APPLICATIONS DISCLAIMER: Atmel products are not designed for and will not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death (“Safety-Critical Applications”) without an Atmel officer's specific written consent. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Atmel products are not designed nor intended for use in military or aerospace applications or environments unless specifically designated by Atmel as military-grade. Atmel products are not designed nor intended for use in automotive applications unless specifically designated by Atmel as automotive-grade.

Downloaded from Arrow.com.