MC68377 Reference Manual

Revised 15 October 2000

Click here for information on how to use the Acrobat full-text Search function.

Click here to go to Table of Contents
Click here to go to List of Figures
Click here to go to List of Tables

PREFACE

This manual describes the capabilities, operation, and functions of the MC68377 microcontroller unit. Documentation for the modular microcontroller family follows the modular construction of the devices in the product line. Each device has a comprehensive user's manual which provides sufficient information for normal operation of the device. The user's manual is supplemented by module reference manuals that provide detailed information about module operation and applications. Please refer to the list of documents below for further information.

Audience

This manual is intended for system software and hardware developers and applications programmers who want to develop products using MC68377. It is assumed that the reader understands operating systems, microprocessor and microcontroller system design.

Additional Reading

The following Motorola documents provide an in-depth functional description of the MC68377 modules:

- ***MC68332 User's Manual*** (**MC68332UM/AD**)
- ***CPU32 Central Processor Unit Reference Manual*** (**CPU32RM/AD**)
- ***QSM Reference Manual*** (**QSMRM/AD**)
- TPU documentation (**TPULITPAK/D**, including the **TPURM/AD**)
- ***MC68336/376 Reference Manual,*** with TouCAN, (**MC68336/376RM/AD**)
- ***Configurable Timer Module Reference Manual*** (**CTMRM/AD**)

Conventions

This document uses the following notational conventions:

ACTIVE_HIGH names for signals that are active high are shown in uppercase text without an overbar. Signals that are active high are referred to as asserted when they are high and negated when they are low.

ACTIVE_LOW shown by a bar over a signal name, indicates that the signal is active low. Active-low signals are referred to as asserted (active) when they are low and negated when they are high.

0x0F shows hexadecimal numbers

0b0011 shows binary numbers

A **specific bit or signal** within a range is referred to by mnemonic and number. For example, ADDR15 is bit 15 of the address bus. A **range of bits or signals** is referred

to by mnemonic and the numbers that define the range. For example, DATA[7:0] form the low byte of the data bus.

REG[FIELD] are abbreviations or acronyms for registers are shown in uppercase text. Specific bit fields or ranges are shown in brackets.

x in certain contexts, such as signal encoding, indicates a don't care. For example, if a field is binary encoded 0bx001, the state of the first bit is a don't care.

NOTE

Throughout this manual references to 3 V refer to the nominal supply voltage of 3.3 V.

Nomenclature

Logic level one is the voltage that corresponds to Boolean true (1) state.

Logic level zero is the voltage that corresponds to Boolean false (0) state.

To **set** a bit or bits means to establish logic level one on the bit or bits.

To **clear** a bit or bits means to establish logic level zero on the bit or bits.

LSB means least significant bit or bits. **MSB** means most significant bit or bits. References to low and high bytes are spelled out.

Asserted means that a signal is in active logic state. An active low signal changes from logic level one to logic level zero when asserted. An active high signal changes from logic level zero to logic level one when asserted.

Negated means that an asserted signal changes logic state. An active low signal changes from logic level zero to logic level one when negated, and an active high signal changes from logic level one to logic level zero.

A range of mnemonics is referred to by mnemonic and the numbers that define the range. For example, VBR[4:0] are bits four to zero of the vector base register.

Symbols

Symbols and Operators

Symbol	Function
+	Addition
–	Subtraction (two's complement) or negation
*	Multiplication
/	Division
>	Greater
<	Less
=	Equal
≥	Equal or greater
≤	Equal or less
≠	Not equal
•	AND
;	Inclusive OR (OR)
⊕	Exclusive OR (EOR)
NOT	Complementation
:	Concatenation
?	Transferred
¤	Exchanges
±	Sign bit; also used to show tolerance
«	Sign extension
%	Binary value
$	Hexadecimal value

References

It is recommended to use the Sematech *Official Dictionary* and the *Reference Guide to Letter Symbols for Semiconductor Devices* by the JEDEC Council/Electronics Industries Association as references for terminology and symbology.

SECTION 1
OVERVIEW

1.1 Introduction

The MC68377 is a member of the MC68300 family of modular microcontrollers. This family includes a series of modules from which numerous microcontrollers (MCUs) are being assembled. These modules are connected on-chip via the inter-module bus (IMB).

A short description of each module used in the MC68377 appears in the following sections.

1.2 Module List

The MC68377 chip contains twelve modules. These modules are:

1. MC68000 family central processing unit **(**CPU32X)
2. Burst integration module **(**BIM)
3. 32-Kbyte (16-Kbyte x 16) fast access SRAM module (FASRAM)
4. Two time processor units (TPU3)
5. Dual-port RAM for TPU3 (DPTRAM)
6. Two queued serial modules (QSM)
7. 10-bit queued analog to digital converter module (QADC64)
8. Analog multiplexer (AMUX)
9. Configurable timer module (CTM9)
10. CAN serial communication module (TouCAN)
11. J1850 serial communication module (DLCMD2)
12. Four 512-byte SRAM modules (SRAM)

1.3 Referenced Documents

As a complement to the present document, the following Motorola documents provide an in-depth functional description of the MC68377 modules:

- MC68332 User's Manual (**MC68332UM/AD**)
- CPU32 Central Processor Unit Reference Manual (**CPU32RM/AD**)
- QSM Reference Manual (**QSMRM/AD**)
- TPU documentation (**TPULITPAK/D**, including the **TPURM/AD**)
- MC68336/376 Reference Manual with TouCAN (**MC68336/376RM/AD**)
- Configurable Timer Module Reference Manual (**CTMRM/AD**)

1.4 Feature List

The major features of the MC68377 are:

- Modular architecture:
  - — Compatible with the current modular library of peripherals
  - — Separate program bus (imb) and data bus (FAB — fast access bus)
  - — Burst mode internal bus (IMB) and external bus for instruction fetches
  - — Burst mode chip select for glueless connection to burst mode memories
  - — Fast one clock access static RAM for data accesses (FASRAM)
  - — Fully static implementation
- 32-Bit 68000 family CPU (CPU32x):
  - — Object code compatible with the CPU32
  - — Greater than 2X performance increase over CPU32 at same system frequency
  - — Clock doubled (2X system clock) operation
  - — Burst mode program fetches
  - — Six word instruction prefetch queue
  - — One-clock fast access data bus (FAB)
  - — Virtual memory implementation
  - — Loop mode of instruction execution
  - — Improved exception handling for controller applications
  - — Table lookup and interpolate instruction
  - — High level language support
  - — Hardware breakpoint signal, background debug mode
- Burst integration module (BIM):
  - — External asynchronous bus and synchronous burst mode bus support
  - — Improved memory access timing
  - — One burst mode chip select output (five pins)
  - — Seven programmable asynchronous chip select outputs
  - — Three modes of operation: master mode, single chip mode, and emulation mode
  - — System protection logic with improved loss-of-oscillator protection
  - — Automatic configuration from shadow registers
  - — System clock based on 5.50-MHz crystal
  - — Watchdog timer, clock monitor, and bus monitor
- 32-Kbyte (16-Kbyte x 16) fast access SRAM (FASRAM):
  - — Dynamic dual access on IMB and FAB
  - — Separate standby power supply pin for 32 Kbytes of standby SRAM
- Two serial I/O subsystems (queued serial module: QSM):
  - — Enhanced SCI (UART): modulus baud rate generator, parity
  - — Queued SPI: 80-byte RAM, up to 16 automatic transfers, continuous cycling, eight to 16 bits per transfer, LSB/MSB first
  - — Dual function I/O port pins
- 10-Bit queued analog to digital converter with internal analog multiplexer (QADC64/AMUX):
  - — 26 channels (with two 8-bit internal analog MUXs)
  - — Four automatic channel selection and conversion modes

- — Two channel scan queues of variable length
- — Queue pointers indicate current location for each queue
- — Automated queue modes initiated by software command, external edge trigger/level gate, or one periodic/interval timer assignable to both queue one and two
- — Sub-queues possible using pause mechanism
- — Queue complete and pause software interrupts available in both queues
- — 64 result registers and three result alignment formats
- — Programmable input sample time
- — 10-bit A/D converter with internal sample/hold
- — Typical conversion time is 10 µs (100-Kbyte samples/sec) full accuracy
- — Programmable frequency and duty cycle for A/D converter clock

• Two time processor units (TPU3) (refer to **Appendix C**):
- — 16 channels — each is associated with a pin
- — Each channel can perform any time function
- — Each time function may be assigned to more than one channel
- — Each channel has an event register comprised of: 16-bit capture register, 16-bit compare/match register, 16-bit ≥ comparator
- — Each channel can be programmed to perform match or capture operations with one, or both, of the two 16-bit free running timer count registers (TCR1 and TCR2)
- — TCR1 is clocked from the internal TPU3 system clock
- — TCR2 may be clocked or gated from the external T2CLK pin
- — All time primitives are microcoded
- — Four Kbytes of microstore program ROM space
- — All channels have eight 16-bit parameter registers
- — A hardware scheduler with three priority levels is included
- — Resolution is one system clock period
- — Modulus prescaler (DIV 2, 4, 6..... 62, 64)

• Six Kbytes dual port memory for TPU3 (DPTRAM):

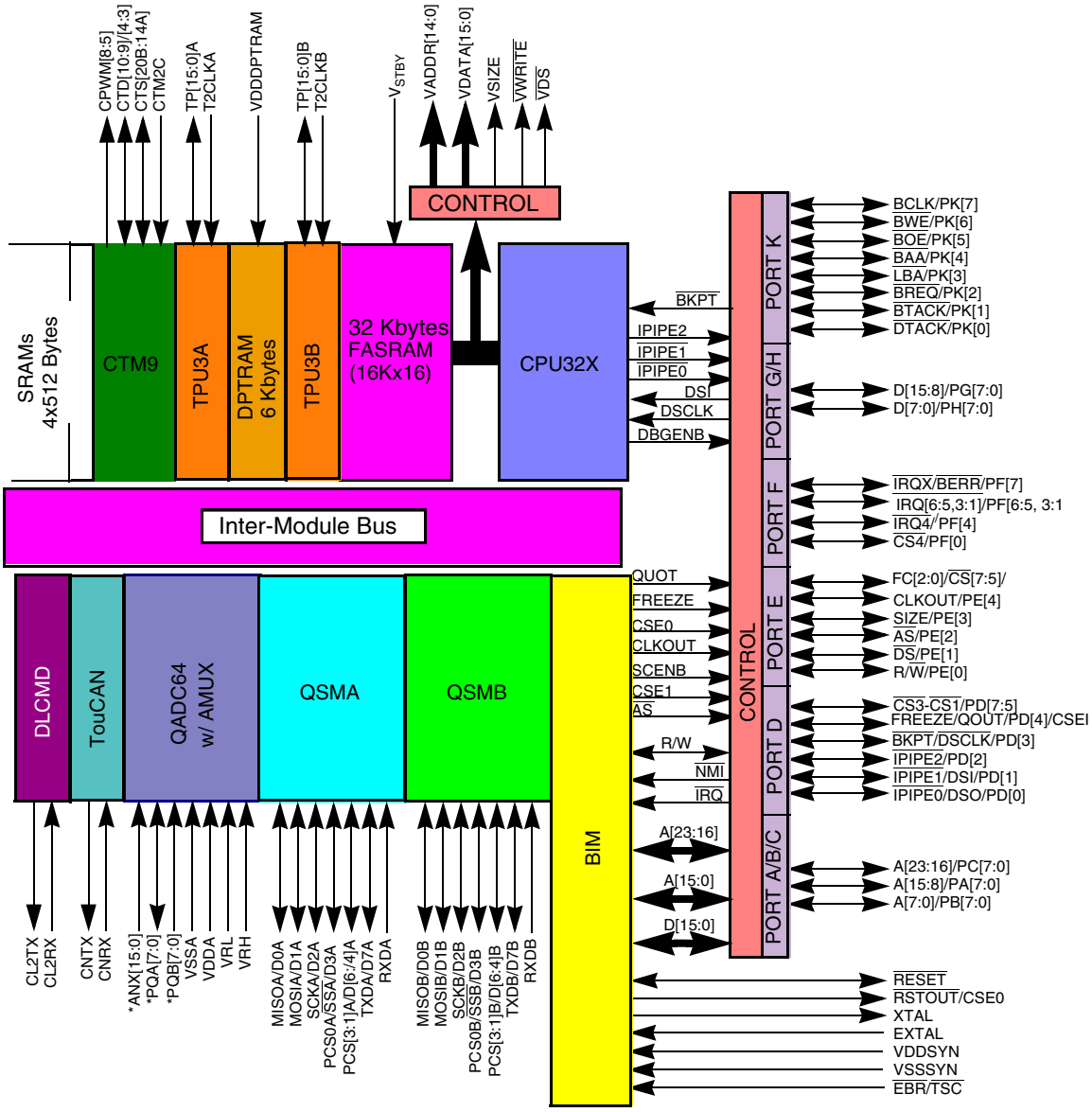
• Configurable timer module #9 (CTM9):
- — One bus interface unit submodule (BIUSM)
- — One counter prescaler submodule (CPSM)
- — One free-running counter submodule (FCSM)
- — Two modulus counter submodule (MCSM)
- — Four single action submodules (SASM)
- — Four double action submodules (DASM)
- — Four dedicated PWM submodules (PWMSM)

• Data link controller module digital (DLCMD)
- — Requires a 22 MHz system clock for correct bit timing
- — GM class-two compatible / SAE J1850 compatible
- — 10.4 KBps VPW bit format
- — Handles all network protocol functions
- — Message buffering on transmit (11 bytes — full message) and on receive (20 bytes)
- — Hardware CRC generation and checking
- — Transmit and receive block mode support

- CAN 2.0B controller module (TouCAN™):
  - — Full implementation of CAN protocol — version 2.0B.
  - — Standard/extended data and remote frames (up to 109/127 bits long).
  - — Programmable bit rate up to one Mbit/s, derived from system clock.
  - — 16 Rx/Tx message buffers of 0-8 bytes data length, of which two buffers are configurable to work as Rx buffers with specific programmable masks.
  - — Full implementation of CAN protocol — version 2.0B.
  - — Standard/extended data and remote frames (up to 109/127 bits long).
  - — Programmable bit rate up to one Mbit/s, derived from system clock.
  - — 16 Rx/Tx message buffers of 0-8 bytes data length, of which two buffers are configurable to work as Rx buffers with specific programmable masks.
- SRAM
  - — Two Kbytes (4 x 512 bytes) static RAM (SRAM).
- Package: 324-pin BGA
- Technology: sub-micron HCMOS
- Operating temperature: -40° C to 125° C
- Operating frequency: 22.00-MHz maximum system clock at V_{DD} = 3.3 V 5%
- 3.3-V core voltage (3.3-V external interface with 5-V tolerant inputs).
- 5-V general purpose digital I/O.

1.5 Functional Block Diagram

A functional block diagram of the MC68377 chip appears in **Figure 1-1**:



*See **Table 8-1** for breakdown of pin functions.

Figure 1-1 Block Diagram

Table 1-1 MC68377 Pin Usage/Pin Definitions

Module	Pin Name	Pins	I/O	Function	Input Voltage	Act Lvi	Output Voltage	Drive Stren.	Driver	Input Buffer	Pull Up/ Down	Output Type
BIM	A[15:8]/PA[7:0]	8	I/O	Address[15:8]	3.3 V	—	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	—	A[1]
			I/O	PortA[7:0]	5 V	—	sel[2]	sel[3]				
	A[7:0]/PB[7:0]	8	I/O	Address[7:0]	3.3 V	—	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	—	A[1]
			I/O	PortB[7:0]	5 V	—	sel[2]	sel[3]				
	A[23:16]/ PC[7:0]	8	I/O	Address[23:16]	3.3 V	—	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	—	A[1]
			I/O	PortB[7:0]	5 V	—	sel[2]	sel[3]				
	IPIPE0/DSO/ PD[0]	1	O	Pipe tracking	—	—	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	—	A
			O	BDM serial output data	—	—	3.3 V	40 pf/80 pf				
			I/O	PortD[0]	5 V	—	sel[2]	sel[3]				
	IPIPE1/DSI/ PD[1]	1	O	Pipe tracking	—	—	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	—	A
			I	BDM serial data input	3.3 V	—	—	—				
			I/O	PortD[1]	5 V	—	sel	sel[3]				
	IPIPE2/PD[2]	1	O	Pipe tracking	—	—	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	—	A
			I/O	PortD[1]	5 V	—	sel[2]	sel[3]				
	$\overline{BKPT}$/$\overline{DSCLK}$/ PD[3]	1	I/O	Break point	3.3 V	L	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	U	A
			I	BDM serial clock	3.3 V	—	—	—				
			I/O	PortD[3]	5 V	—	sel[2]	sel[3]				
	FREEZE/ QUOT/CSE1/ PD[4]	1	O	Freeze	—	H	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	TP	A
			O	Quotient out	—	—	3.3 V	40 pf/80 pf				
			O	Emulation mode chip select[1]	—	—	3.3 V	40 pf/80 pf				
			I/O	PortD[4]	5 V	—	sel[2]	sel[3]				
	$\overline{CS}$[3:1]/PD[7:5]	3	O	Chip selects[3:1]	—	L	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	TP	A[1]
			I/O	PortD[7:5]	5 V	—	sel[2]	sel[3]				
	R/$\overline{W}$/PE[0]	1	I/O	Read/write	—	L	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	TP	A[1]
			I/O	PortE[0]	5 V	—	sel[2]	sel[3]				
	$\overline{DS}$/PE[1]	1	I/O	Data strobe	3.3 V	L	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	TP	A[1]
			I/O	PortE[1]	5 V	—	sel[2]	sel[3]				
	$\overline{AS}$/PE[2]	1	I/O	Address strobe	3.3 V	L	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	TP	A[1]
			I/O	PortE[2]	5 V	—	sel[2]	sel[3]				
	SIZE/PE[3]	1	I/O	Transfer size	3.3 V	—	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	U/D[4]	A[1]
			I/O	PortE[3]	5 V	—	sel[2]	sel[3]				
	CLKOUT/PE[4]	1	O	Clock output	—	—	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	—	A
			I/O	PortE[2]	5 V	—	sel[2]	sel[3]				

Table 1-1 MC68377 Pin Usage/Pin Definitions (Continued)

Module	Pin Name	Pins	I/O	Function	Input Voltage	Act Lvi	Output Volt-age	Drive Stren.	Driver	Input Buffer	Pull Up/ Down	Out-put Type
BIM	FC[2:0]/ $\overline{CS}$[7:5]/PE[7:5]	3	O	Function code[2:0]	—	—	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	—	A[1]
			O	Chip selects[7:5]	—	L	3.3 V	40 pf/80 pf				
			I/O	PE[7:5]	5 V	—	sel[2]	sel[3]				
	$\overline{CS}$[4]/PF[0]	1	O	Chip select[4]	—	L	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	—	A[1]
			I/O	PortF[0]	5 V	—	sel[2]	sel[3]				
	$\overline{IRQ}$[3:1]/ PF[3:1]	3	I/O	Interrupt request[3:1]	3.3 V	L	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	—	A
			I/O	PortF[3:1]	5 V	—	sel[2]	sel[3]				
	$\overline{IRQ4}$/PF[4]	1	I/O	Interrupt request[4]	3.3 V	L	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	—	A
			I/O	PortF[4]	5 V	—	sel[2]	sel[3]				
	$\overline{IRQ}$[6:5]/ PF[6:5]	2	I/O	Interrupt request[6:5]	3.3 V	L	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	—	A
			I/O	PortF[6:5]	5 V	—	sel[2]	sel[3]				
	$\overline{IRQX}$/$\overline{BERR}$/ PF[7]	1	I/O	Interrupt request	3.3 V	L	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	U	A
			I/O	Bus Error	3.3 V	L	sel[2]	3.3 V				
			I/O	PortF[7]	5 V	—	sel[2]	sel[3]				
	D[11,9,8]/ PG[3:,1,0]	3	I/O	Data bus[11,9,8]	3.3 V	—	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	U/D[5]	A[1]
			I/O	PortG[3,1,0]	5 V	—	sel[2]	3.3 V				
	D[15:12,10]/ PG[7:4,]	5	I/O	Data bus[15:12,10]	3.3 V	—	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	U	A[1]
			I/O	PortG[7:4,2]	5 V	—	sel[2]	3.3 V				
	D[7:2,0]/ PH[7:2,0]	7	I/O	Data bus[7:0]	3.3 V	—	3.3 V	40 pf/80 pf	TP	Sync	U/D[5]	A[1]
			I/O	PortH[7:0]	5 V	—	sel[2]	3.3 V				
	D[1]/PH[1]	1	I/O	Data bus[1]	3.3 V	—	3.3 V	40 pf/80 pf	TP	Sync	U	A[1]
			I/O	PortH[1]	5 V	—	sel[2]	3.3 V				
	$\overline{DTACK}$/PK[0]	1	I/O	Data bus[1]	3.3 V	—	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	U/D[4]	A
			I/O	PortH[1]	5 V	—	sel[2]	3.3 V				
	$\overline{BTACK}$/PK[1]	1	I/O	Burst transfer acknowledge	3.3 V	L	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	U/D[4]	A
			I/O	PortK[1]	5 V	—	sel[2]	3.3 V				
	$\overline{BREQ}$/PK[2]	1	I/O	Burst request	3.3 V	L	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	U/D[4]	A[1]
			I/O	PortK[2]	5 V	—	sel[2]	3.3 V				
	$\overline{LBA}$/PK[3]	1	O	Burst CS load burst address	—	L	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	U/D[4]	A[1]
			I/O	PortK[3]	5 V	—	sel[2]	3.3 V				
	$\overline{BAA}$/PK[4]	1	O	Burst CS ad-dress advance	—	L	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	—	A[1]
			I/O	PortK[4]	5 V	—	sel[2]	3.3 V				

Table 1-1 MC68377 Pin Usage/Pin Definitions (Continued)

Module	Pin Name	Pins	I/O	Function	Input Voltage	Act Lvi	Output Volt-age	Drive Stren.	Driver	Input Buffer	Pull Up/ Down	Out-put Type
BIM	$\overline{BOE}$/PK[5]	1	O	Burst CS output enable	—	L	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	U/D[4]	A[1]
			I/O	PortK[5]	5 V	—	sel[2]	3.3 V				
	$\overline{BWE}$/PK[6]	1	O	Burst CS write enable	—	L	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	—	A[1]
			I/O	PortK[6]	5 V	—	sel[2]	3.3 V				
	BCLK/PK[7]	1	O	Burst CS clock	—	—	3.3 V	40 pf/80 pf	TP	Hyst/ Sync	—	A[1]
			I/O	PortK[7]	5 V	—	sel[2]	3.3 V				
	RESET	1	I/O	Reset	3.3 V	L	3.3 V	40 pf/80 pf	OD	Hyst/ Sync	U	A
	$\overline{RSTOUT}$/CSE0	1	O	Burst CS clock	—	L	3.3 V	40 pf/80 pf	TP	—	U	A[6]
			O	PortK[7]	—	—	3.3 V	40 pf/80 pf				
	XTAL	1	—	Crystal	—	L	—	—	—	—	—	—
	EXTAL	1	—	Crystal external clock	—	L	—	—	—	—	—	—
	$\overline{EBR}$/$\overline{TSC}$	1	I	External bus request	3.3 V	L	—	—	—	Hyst/ Sync	U	A
			I	Tri-state control	3.3 V	L	—	—				
	Total for Module = 77 pins											
FAS-RAM	VADDR[14:0]	15	O	Visibility ad-dress bus	—	—	3.3 V	40 pf/80 pf	TP	—	D	A
	VDATE[15:0]	16	O	Visibility data bus	—	—	3.3 V	40 pf/80 pf	TP	—	D	A
	VWRITE	1	O	Visibility bus write strobe	—	L	3.3 V	40 pf/80 pf	TP	—	D	A
	$\overline{VDS}$	1	O	Visibility bus data strobe	—	L	3.3 V	40 pf/80 pf	TP	—	D	A
	VSIZE	1	O	Visibility bus operand size	—	—	3.3 V	40 pf/80 pf	TP	—	D	A
	V_{STBY}	1	—	Standby supply voltage	3.3 V	—	—	—	—	—	—	—
	Total For Module = 35 Pins											

Table 1-1 MC68377 Pin Usage/Pin Definitions (Continued)

Module	Pin Name	Pins	I/O	Function	Input Voltage	Act Lvi	Output Volt-age	Drive Stren.	Driver	Input Buffer	Pull Up/ Down	Out-put Type
QADC64	AN[54:52]/ PQA[2:0]	3	I	Port A analog inputs[54:52]	An	—	—	—	—	—	—	B
			I/O	Port A digital I/O[2:0]	5 V	—	5 V/ sel[7]	50 pf	TP	Hyst/ Sync		
	AN55/ETRIG1/ PQA3	1	I	Port A analog input[55]	An	—	—	—	—	—	—	B
			I/O	Port A digital I/O[3]	5 V	—	5 V/ sel[7]	50 pf	TP	Hyst/ Sync		
	AN56/ETRIG2/ PQA4	1	I	Port A analog input[56]	An	—	—	—	—	—	—	B
			I	Port A external trigger[2]	5 V	L/H	—	—	—	Hyst/ Sync		
			I/O	Port A digital I/O[4]	5 V	—	5 V/ sel[7]	50 pf	TP	Hyst/ Sync		

Table 1-1 MC68377 Pin Usage/Pin Definitions (Continued)

Module	Pin Name	Pins	I/O	Function	Input Voltage	Act Lvi	Output Voltage	Drive Stren.	Driver	Input Buffer	Pull Up/ Down	Output Type
QADC64	AN[59:57]/ PQA[7:5]	3	I	Port A analog inputs[59:57]	An	—	—	—	—	—	—	B
			I/O	Port A digital I/O[7:5]	5 V	—	5 V/ sel[7]	50 pf	TP	Hyst/ Sync		
	AN0/ANW/ PQB0 (For Test Purposes Only)	1	I	Por tB analog input[0]	An	—	—	—	—	—	—	B
			I	Port B analog input[W]	An	—	—	—	—	—		
			I	Port B digital input[0]	5 V	—	—	—	—	Hyst/ Sync		
	AN1/ANX/ PQB1 (For Test Purposes Only)	1	I	PortB analog input[1]	An	—	—	—	—	—	—	B
			I	Port B analog input[X]	An	—	—	—	—	—		
			I	Port B digital input[1]	5 V	—	—	—	—	Hyst/ Sync		
	AN2/ANY/ PQB2 (For Test Purposes Only)	1	I	Port B analog input[2]	An	—	—	—	—	—	—	B
			I	Port B analog input[Y]	An	—	—	—	—	—		
			I	Port B digital input[2]	5 V	—	—	—	—	Hyst/ Sync		
	AN3/ANZ/ PQB3	1	I	Port B analog input[3]	An	—	—	—	—	—	—	B
			I	Port B analog input[Z]	An	—	—	—	—	—		
			I	Port B digital input[3]	5 V	—	—	—	—	Hyst/ Sync		
	AN[51:48]/ PQB[7:4]	4	I	Port B analog inputs[51:48]	An	—	—	—	—	—	—	B
			I	Port B digital inputs[7:4]	5 V	—	—	—	—	Hyst/ Sync		
	ANX[15:0]	16	I	Mux'd analog inputs[15:0]	An	—	—	—	—	—		B
	V_{RH}	1	—	Voltagle reference high	5 V	—	—	—	—	—		—
	V_{RL}	1	—	Voltagle reference low	—	—	—	—	—	—		—
	V_{DDA}	1	—	Analog supply	5 V	—	—	—	—	—		—
	V_{SSA}	1	—	Analog ground	—	—	—	—	—	—		—
	Total For Module = 36 Pins											

Table 1-1 MC68377 Pin Usage/Pin Definitions (Continued)

Module	Pin Name	Pins	I/O	Function	Input Voltage	Act Lvi	Output Volt-age	Drive Stren.	Driver	Input Buffer	Pull Up/ Down	Out-put Type
QSMA	MISOA/D0A	1	I/O	Master in/slave out	5 V	—	5 V/50 ns	200 pf	TP[8]	Hyst/ Sync	—	C
			I/O	General purpose I/O D0	5 V	—	5 V/ 600 ns	50 pf		Hyst		
	MOSIA/D1A	1	I/O	Master out/slave in	5 V	—	5 V/50 ns	200 pf	TP[8]	Hyst/ Sync	—	C
			I/O	General purpose I/O D1	5 V	—	5 V/ 600 ns	50 pf		Hyst		
	PCS0A/$\overline{\text{SSA}}$/ D3A	1	I/O	Peripheral chip select0	5 V	—	5 V/50 ns	200 pf	TP[8]	Hyst/ Sync	—	C
			I	Slave select	5 V	L	—	—		Hys/ Sync		
			I/O	General purpose I/O D3	5 V	—	5 V/ 600 ns	50 pf		Hyst		
	PCS[3:1]A/ D[6:4]A	3	I/O	Peripheral chip selects[3:1]	5 V	—	5 V/50 ns	200 pf	TP[8]	Hyst/ Sync	—	C
			I/O	General purpose I/O D[6:4]	5 V	—	5 V/ 600 ns	50 pf		Hyst		
	TxDA/D7A	1	O	SCI transmit date	—	—	5 V/50 ns	200 pf	TP[8]	—	—	C
			I/O	General purpose I/O D7	5 V	—	5 V/ 600 ns	50 pf		Hyst		
	RxDA	1	I	SCI receive data	5 V	—	—	—		Hyst		
	Total For Module = 9 pins											

Table 1-1 MC68377 Pin Usage/Pin Definitions (Continued)

Module	Pin Name	Pins	I/O	Function	Input Voltage	Act Lvi	Output Volt-age	Drive Stren.	Driver	Input Buffer	Pull Up/ Down	Out-put Type
QSMB	MISOB/D0B	1	I/O	Master in/slave out	5 V	—	5 V/50 ns	200 pf	TP[8]	Hyst/ Sync	—	C
			I/O	General purpose I/O D0	5 V	—	5 V/ 600 ns	50 pf		Hyst		
	MOSIB/D1B	1	I/O	Master out/slave in	5 V	—	5 V/50 ns	200 pf	TP[8]	Hyst/ Sync	—	C
			I/O	General purpose I/O D1	5 V	—	5 V/ 600 ns	50 pf		Hyst		
	PCS0B/$\overline{\text{SSB}}$/ D3B	1	I/O	Peripheral chip select0	5 V	—	5 V/50 ns	200 pf	TP[8]	Hyst/ Sync	—	C
			I	Slave select	5 V	L	—	—		Hys/ Sync		
			I/O	General purpose I/O D3	5 V	—	5 V/ 600 ns	50 pf		Hyst		
	PCS[3:1]B/ D[6:4]B	3	I/O	Peripheral chip selects[3:1]	5 V	—	5 V/50 ns	200 pf	TP[8]	Hyst/ Sync	—	C
			I/O	General purpose I/O D[6:4]	5 V	—	5 V/ 600 ns	50 pf		Hyst		
	TxDB/D7B	1	O	SCI transmit date	—	—	5 V/50 ns	200 pf	TP[8]	—	—	C
			I/O	General purpose I/O D7	5 V	—	5 V/ 600 ns	50 pf		Hyst		
	RxDB	1	I	SCI receive data	5 V	—	—	—		Hyst		
	Total For Module = 9 pins											
TPU3A	TP[15:0]A	16	I/O	Timer channel	5 V	L/H	5 V/ sel[7]	50 pf	TP	Hyst	—	C
	T2CLKA	1	I	External clock input	5 V	L/H	—	—	—	Hyst	U	C
	Total For Module = 17 pins											
TPU3B	TP[15:0]B	16	I/O	Timer channel	5 V	L/H	5 V/ sel[7]	50 pf	TP	Hyst	—	C
	T2CLKB	1	I	External clock input	5 V	L/H	—	—	—	Hyst	U	C
	Total For Module = 17 pins											
CTM9	CTM2C	1	I	External clock input	5 V	L/H	—	—	—	Hyst/ Sync	—	C
	CTD[4:3]	2	I/O	DASM channels[4:3]	5 V	L/H	5 V/ sel[7]	50 fp	TP	Hyst/ Sync	—	C
	CTM[8:5]	4	I/O	PWMSM channels[8:5]	5 V	L/H	5 V/ sel[7]	50 fp	TP	—	—	C
	CTD[10:9]	2	I/O	DASM Channels[10:9]	5 V	L/H	5 V/ sel[7]	50 fp	TP	Hyst/ Sync	—	C
	CTS[20B:14A]	8	I/O	SASM channels[20:14]	5 V	L/H	5 V/ sel[7]	50 fp	TP	Hyst/ Sync	—	C
	Total For Module = 17 pins											

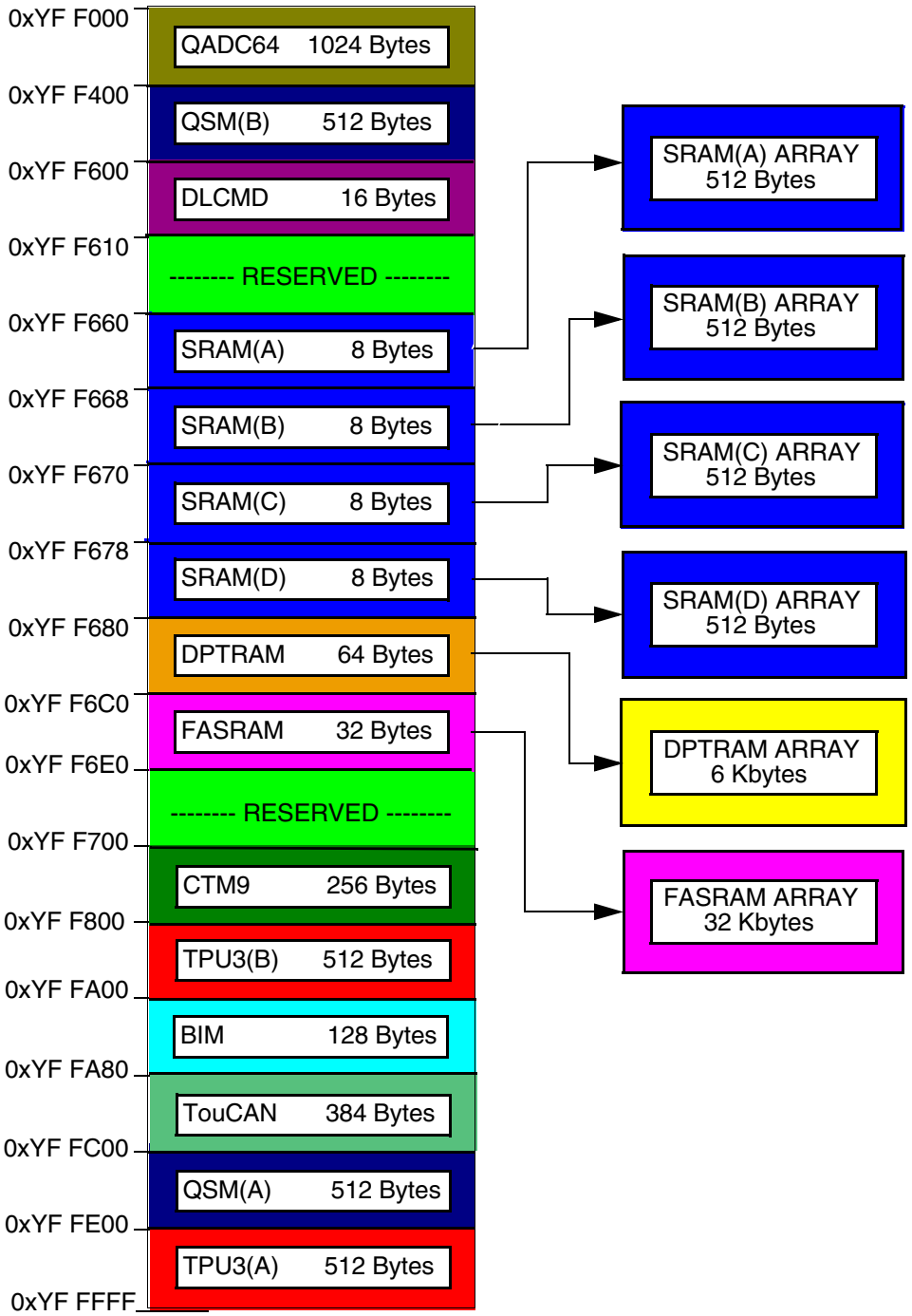
Table 1-1 MC68377 Pin Usage/Pin Definitions (Continued)

Module	Pin Name	Pins	I/O	Function	Input Voltage	Act Lvi	Output Volt-age	Drive Stren.	Driver	Input Buffer	Pull Up/ Down	Out-put Type
Tou-CAN	CNRX	1	I	CAN receive	5 V	—	—	—	—	—	—	C
	CNTX	1	O	CAN transmit	5 V	—	5 V/ 50ns	200 pf	TP[8]	—	—	C
	Total For Module = 2 pins											
DLMCD	CL2RX	1	I	Class 2 receive	5 V	—	—	—	—	—	—	C
	CL2TX	1	O	Class 2 transmit	5 V	—	5 V/50 ns	50 pf	TP	—	—	C
	CNRX	1	I	CAN receive	5 V	—	—	—	—	—	—	C
	Total For Module = 2 pins											

Table 1-1 MC68377 Pin Usage/Pin Definitions (Continued)

Module	Pin Name	Pins	I/O	Function	Input Voltage	Act Lvi	Output Volt-age	Drive Stren.	Driver	Input Buffer	Pull Up/ Down	Out-put Type
POW-ER/ GND	V_{DDSYN}	1	—	Clock synthesiz-er power	3.3 V	—	—	—	—	—	—	—
	V_{SSSYN}	1	—	Clock synthesiz-er ground	—	—	—	—	—	—	—	—
	V_{DD3}	73	—	Supply voltage	3.3 V	—	—	—	—	—	—	—
	V_{SS}	23	—	Ground	—	—	—	—	—	—	—	—
	V_{DDS5}	10	—	Supply voltage	5 V	—	—	—	—	—	—	—
	$V_{DDPTRAM}$	1	—	DPTRAM supply	3.3 V	—	—	—	—	—	—	—
	Power/Ground = 103 pins											
Total Number of Pins for MCU = 324 Pins												
Sync = Synchronized Hyst = Hysteresis OD = Open Drain TP = Totem Pole An = Analog												

NOTES:
1. Driver three-stated on EBR.
2. Either 5-V / 600-ns slew rate or 3.3-V no slew rate control based upon state of FASTIO bit in BIM MCR.
3. Either 50 pf or 40 pf/80 pf drive strength based upon state of FASTIO bit in BIM MCR.
4. Pullup or pulldown at RESET depending upon PCON register bit. Disabled after RESET.
5. Pullup or pulldown at RESET depending upon PCON register bit. Pullup after RESET.
6. Will not three-state on TSC (required by PLL test mode).
7. Either 5-V / 600-ns or 5-V / 50-ns slew rate based upon state of FASTIO bit in BIM MCR.
8. Module will configure signals for OD operation.



Y = M111, where M is the MODMAP signal state on the IMB, which reflects the state of the MODMAP in the module configuration register of the burst integration module. (Y= 0x7 or 0xF).

Figure 1-2 MC68377 Address Map

SECTION 2
CPU32X

This document describes the modifications of the CPU32 to create the CPU32X.

2.1 Features

The CPU32X is greater than two times faster at the same external clock rate than the CPU32 using similar speed memory devices. This performance improvement is obtained by using burst mode memory for instruction fetches, an instruction queue to buffer the burst instruction fetches, one clock operand (data) accesses on the fast access bus and by increasing the speed of the CPU.

NOTE

All references to clocks are to the external clock.

The CPU32X and CPU32 instruction sets are identical. Interrupts, breakpoints and bus errors function exactly the same in the CPU32X as they do in the CPU32 except with reduced response time. Retry, relinquish and retry (RRT), and late bus error are not supported by the CPU32X. Please refer to the ***CPU32 Reference Manual (CPU32RM/AD)***, for the instruction set, interrupt operation, breakpoint operation, bus error operation and programmer's model.

The CPU32X burst mode bus protocol is an extension of the intermodule bus (IMB) protocol. The CPU32X will support all of the existing and future IMB peripherals. Burst mode IMB program memory (EEPROM, flash EPROM, ROM) can be implemented on the IMB to support the CPU32X. The external burst IMB protocol is supported by the burst mode system integration module (BIM).

The fast access bus is connected to the fast access static RAM (FASRAM) and increases the data fetch bandwidth of the CPU32X by reducing the minimum data access time to one clock. In addition the fast access bus separates data accesses from program accesses such that if the data being accessed is in the FASRAM the data will be returned to the CPU in one clock and the IMB burst mode program fetch can continue without interruption.

2.2 CPU32X System Architecture

The recommended system architecture of the CPU32X includes a fast access standby random access memory (FASRAM) and requires a burst mode interface module (BIM). Refer to **Figure 2-1** for the block diagram of a CPU32X-based device.

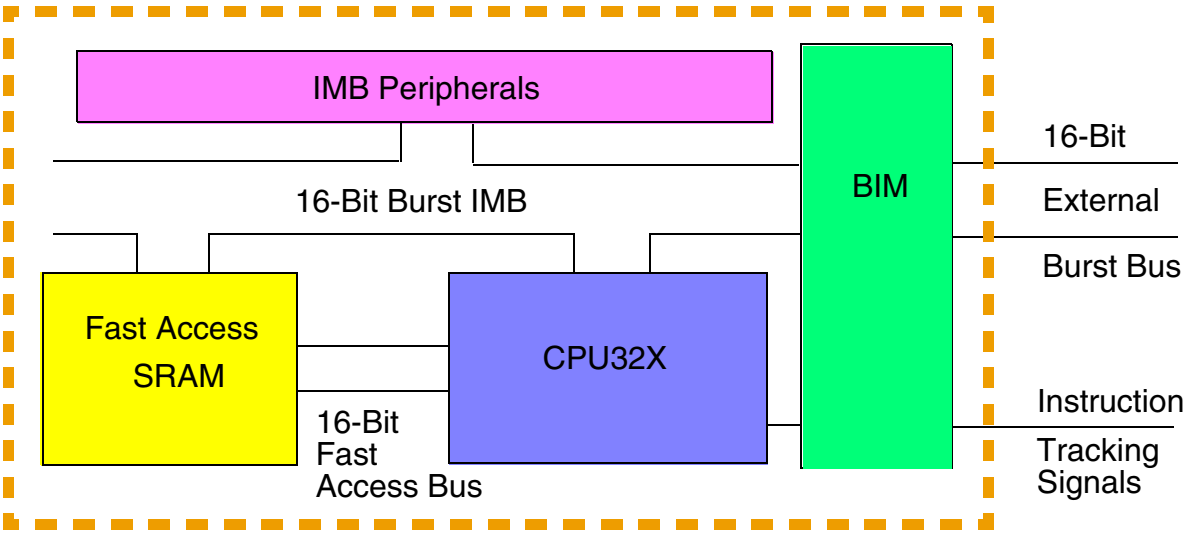


Figure 2-1 System Architecture

2.3 Optimizing System Performance

Data accesses to the fast access SRAM optimize the performance of the CPU32X system therefore the stack and the most frequently used variables should be located in the FASRAM. Accesses in the FASRAM are one clock while accesses outside the FASRAM normally require two clocks plus the number of clocks required to run the IMB or external access. For example the access time of a two-clock IMB SRAM would be four clocks total (two clocks of internal delay plus two clocks for IMB SRAM access). The normal total access time for any IMB based memory, IMB based peripheral or external device is four clocks. (Two clocks delay from the CPU32X and two clocks for the fastest possible bus cycle). If IMB or external bus cycles are longer than two clocks the bus cycle will be extended. If for some reason the IMB is idle then the minimum number of clocks for a non-FASRAM access is three clocks. The IMB can be idle due to a full prefetch queue or for other reasons. Refer to **Table 2-2** for an example of a data access to the FASRAM. Refer to **Table 2-3** for an example of a data access not in the FASRAM.

2.4 Instruction Execution

This section describes instruction execution on the CPU32X. External clock cycles are used to provide as accurate as possible operation and timing of the instruction examples. Because exact execution time for an instruction or operation depends on independently scheduled resources, on memory speeds, and on other variables these diagrams may not exactly match the CPU32X operation in every application.

2.4.1 Resource Scheduling

The CPU32X contains several independently scheduled resources. The organization of these resources within the CPU32X is shown in **Figure 2-2**. Some variation in

instruction execution timing results from concurrent resource utilization. Due to these concurrent resources exact instruction execution times are difficult to predict until the instruction sequence is actually executed. Identical sequences of instructions and memory accesses will not vary in execution time or bus access pattern from one execution of the identical sequence to another.

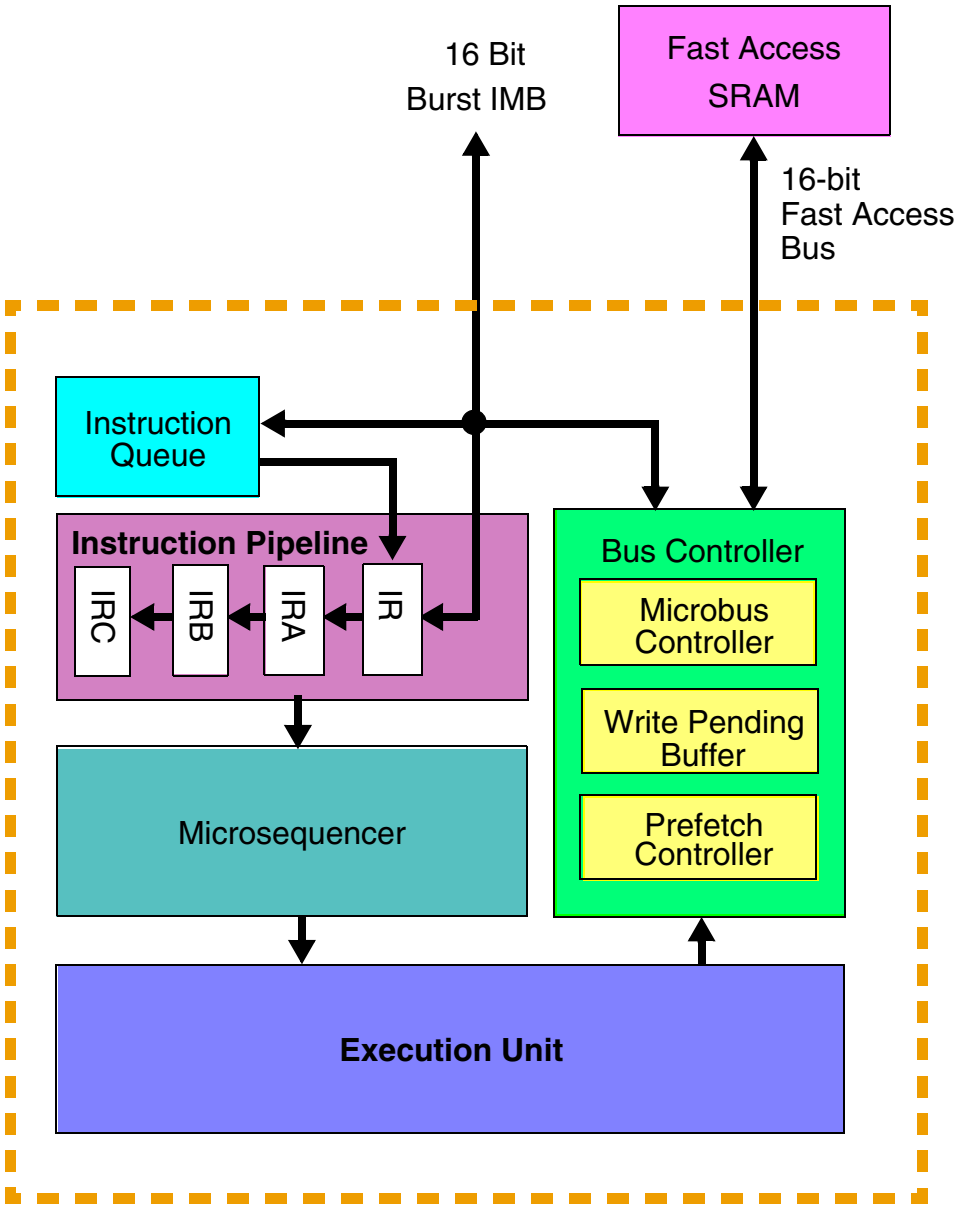


Figure 2-2 CPU32X Internal Architecture

2.4.2 Microsequencer

The microsequencer either executes microinstructions or awaits completion of accesses necessary to continue microcode execution. The microsequencer supervises the bus controller, instruction execution, and internal processor operations such

as the calculation of an effective address and the setting of condition codes. It also initiates instruction word prefetches after a change of flow and controls validation of instruction words in the instruction pipeline. The microsequencer is the same on the CPU32 as on the CPU32X.

2.4.3 Instruction Pipeline

The CPU32X contains a four-word instruction pipeline where instruction opcodes are decoded. The pipeline operates in two stages: IRA and IRC. Each stage of the pipeline is initially filled under microsequencer control and subsequently refilled by the prefetch controller as it empties.

The IR word of the instruction pipeline is a buffer. IR receives instructions from either the instruction queue or directly from the intermodule bus. This register holds the instruction word until it is emptied by IRA. Instruction words (instruction operation words and all extension words) are decoded at stage IRA. IRB is an intermediate holding register for IRC and is filled from IRA when IRC empties. IRC is filled from IRB. Residual decoding and execution take place in stage IRC.

The instruction pipeline registers IRA, IRB, IRC are the same as on the CPU32. The IR registers and the prefetch equations will require modification to support the CPU32X instruction queue and burst mode bus.

2.4.4 Instruction Queue

The instruction queue is a six word first-in-first-out buffer. The instruction queue is filled by the prefetch controller when the IR word of the pipeline is full. When IR empties, the instruction queue provides the next instruction word or if the instruction queue is empty, the instruction word will come directly off the IMB and into IR. The instruction queue is dual ported such that the prefetch controller can fill the queue as the pipe pulls instructions or extension words out of the queue.

Each word in the queue has a bus error and breakpoint status bit which indicates that the word in that stage was loaded with an instruction or extension word from a bus cycle that encountered either a bus error or a breakpoint. This status is either transferred to the pipe when the word is used in the pipe or they are flushed on a change of flow.

The instruction queue is new on the CPU32X.

2.5 Bus Controller Resources

The bus controller consists of the instruction prefetch controller, the write-pending buffer and the microbus controller. These resources transact all reads, writes, and instruction prefetches required for instruction execution.

The bus controller and microsequencer operate concurrently. The bus controller schedules bus cycles to the IMB and the FASRAM. The bus controller can schedule an IMB prefetch in parallel with a FASRAM access or an IMB access. While the bus

controller is running bus cycles the microsequencer controls effective address calculation or sets condition codes.

The microsequencer can also request a bus cycle. If the data resides in the FASRAM the bus controller completes the bus cycle to the FASRAM, see **Table 2-2**. If the data is not in the FASRAM the bus controller queues the data cycle to run on the IMB. The bus controller then terminates any prefetches in progress and runs the cycle when the current cycle is complete. See **Table 2-3** for more information. Once the bus cycle completes, data is returned to the execution unit. If the first word of a long-word access is not in the FASRAM the microsequencer will run the second word of the long-word access on the IMB bypassing the FASRAM access for the second word. If the second word is in the FASRAM the data will be accessed over the IMB. This is done to improve performance of long-word accesses on the IMB.

2.5.1 Prefetch Controller

The instruction prefetch controller receives an initial request from the microsequencer to initiate burst mode instruction prefetching at a given address. Subsequent burst mode prefetches are initiated or continued by the prefetch controller whenever the instruction queue contains only a few instruction words. Burst prefetching begins as soon as the bus is free of operand accesses previously requested by the microsequencer. Additional state information permits the controller to inhibit or terminate prefetch bursts when a change in instruction flow is executing thereby improving the change of flow performance.

EXAMPLE

Change in flow instructions are bcc, jsr, rts and jmp instructions.

In a typical program, 10 to 25 percent of the instructions cause a change of flow. Each time a change occurs, the instruction queue and instruction pipeline must be flushed and refilled from the new instruction stream. The prefetch controller and microsequencer optimize change of flow instructions by detecting the change of flow early, then terminating prefetches in progress or inhibiting unnecessary prefetches from occurring. For non-change of flow instructions the prefetch controller will schedule prefetches when adequate room is available in the queue.

2.5.2 Write-Pending Buffer

The CPU32X incorporates a single-operand write-pending buffer. The buffer permits the microsequencer to continue execution after a request for a write cycle is queued in the bus controller. The time needed for a write at the end of an instruction can overlap the beginning of the following instruction, and thus reduce overall execution time. Interlocks prevent the microsequencer from overwriting the buffer.

The write pending buffer is the same on the CPU32 as on the CPU32X.

2.5.3 Microbus Controller

The microbus controller performs bus cycles issued by the microsequencer. Operand accesses always have priority over instruction prefetches. Word and byte operands

are accessed in a single CPU-initiated bus cycle, although the external bus interface will be required to initiate a second cycle when a word operand is sent to a byte-sized external port. Long operands are accessed in two bus cycles, most significant word first.

2.6 Execution Unit

The execution unit contains all of the logic for carrying out instruction execution. This logic includes the user visible registers (A0-A7, D0-D7, SP, PC, CCR, etc.), the arithmetic logic unit, the shifter and other functional units required for executing all instructions. In addition the execution unit has access to the pipe for extension words and to the data bus buffers for reading and writing memory. The execution unit performs its operations under the control of the microsequencer.

The execution unit is the same on the CPU32 as on the CPU32X.

2.7 Instruction Execution Examples

The following are instruction execution examples to illustrate three points. First, the overlap of current instruction execution, next instruction decode and burst instruction prefetch. Second, is the interaction between operand bus cycles and the prefetching of instructions. Third, restarting the pipe, microsequencer and instruction prefetching after a change of flow instruction.

These instruction examples represent the expected system implementation. The external burst memory is a 2,1,1,1 burst memory which results in 3,1,1,1 IMB burst accesses. Accesses to IMB based peripherals and external memory are assumed to be two clock accesses. Accesses to the FASRAM are one clock accesses.

2.7.1 Execution Overlap

Table 2-1 is the instruction stream I0, I1 and I2. I0, I1 and I2 are one clock instructions with decode, prefetch and execute operations overlapping. This figure begins with the instruction queue containing instructions I4 and I5, a prefetch is initiated for I6 and the pipe is full with instruction I0 executing.

Table 2-1 Execution Overlap

Clock	1	2	3	4
IMB bus controller	Burst prefetch I6,....			I7
FASRAM	—			
Instruction queue	I4,I5	I5	—	
IR	I3	I4	I5	I6
IRA-decode	I2	I3	I4	I5
IRB	I1	I2	I3	I4
IRC-execute	I0	I1	I2	I3

2.8 Operand Accesses

There are two types of operand accesses (data). An operand access to the FASRAM and an operand access not to the FASRAM. All operand accesses, except IACK cycles, go to the FASRAM. IACK cycles are always run on the IMB.) If the operand is in the FASRAM it is returned in one clock, the instruction continues and burst prefetching continues. If the operand is not in the FASRAM the bus controller terminates the burst prefetch in progress and then runs the operand access cycle on the IMB. Once the operand access cycle completes on the IMB burst instruction prefetching is restarted and the instruction continues. An example of a move instruction accessing the FASRAM is shown in **Table 2-2**. An example of a move instruction to an operand not in the FASRAM is shown in **Table 2-3**. The instruction execution time for the move from the FASRAM is three clocks. The best case instruction execution time for the move from two-clock non-FASRAM memory is five clocks. The worst case instruction execution time for the move from two clock non-FASRAM memory is six clocks. For comparison this same move from two clock memory on the CPU32 requires six clocks.

2.8.1 Move (A0),D0 A0 = FASRAM

Table 2-2 is an example of an access to the FASRAM which occurs in parallel to instruction prefetch. Register A0 points to an address which resides in the FASRAM. In this example instructions I1 through I5 follow the move. The pipe is full and the instruction queue is as shown.

Table 2-2 Move (A0),D0 A0 = FASRAM

Clock	1	2	3	4
IMB bus controller	Burst prefetch I6,...			I7
FASRAM		Read (A0)		
Instruction queue	I4,I5	I4,I5	I4,I5	I5,I6
IR	I3	I3	I3	I4
IRA-decode	I2	I2	I2	I3
IRB	I1	I1	I1	I2
IRC-execute	move (A0),D0			I1

2.9 Move (A0),D0 A0! = FASRAM

Table 2-3 is an example of a move (A0),D0 and A0 points outside the FASRAM. In this example instructions I1 through I5 follow the move. The pipe is full, burst prefetch is in progress and the instruction queue is full as shown. I1, I2 and I3 are one clock instructions. This example is the worst case delay. If the IMB is idle, no burst in progress, the move will execute in five clocks instead of six.

Table 2-3 Move (A0),D0 A0! = FASRAM

Clock	1	2	3	4	5	6	7	8	9
IMB Bus Controller	I4	I5	I6	Read (A0)		Burst Prefetch I7,...			I8
FASRAM		Check (A0)							
Instruction Queue		I4	I4,I5	I4-6			I5,I6	I6	I7
IR	I3						I4	I5	I6
IRA-Decode	I2						I3	I4	I5
IRB	I1						I2	I3	I4
IRC-Execute	move (A0),D0						I1	I2	I3

2.10 Burst Start-up

Table 2-4 is an example of a JMP to instruction I0. The pipe and instruction queue is flushed. The jump instruction is waiting on the pipe to be refilled to complete.

Table 2-4 JMP to I0

Clock	1	2	3	4	5	6	7
IMB bus controller		Burst prefetch I0...			I1	I2	I3
FASRAM							
Instruction queue							
IR							
IRA-decode					I0	I1	I2
IRB						I0	I1
IRC-execute	JMP to I0						I0

2.11 Exceptions

Interrupt, breakpoint and bus error exceptions function the same on the CPU32X as on the CPU32. Retry and relinquish and retry (RRT) are not supported. The late bus termination is also not supported. Please refer to the Section 6 of the ***CPU32 User's Manual*, CPU32RM/AD** for information on how the CPU32X interrupt and bus error exceptions function.

2.11.1 Standard Bus Cycle Exceptions

Standard (non-burst) IACK breakpoint and bus error exception bus cycles operate the same on the CPU32X as on the CPU32. In addition interrupts and IMB peripheral breakpoints operate the same as on the CPU32. Retry, RRT and late bus error are supported by the CPU32 but are NOT supported on the CPU32X.

2.11.2 Burst Bus Cycle Exceptions

Breakpoint and bus error exception bus cycles are extended as follows to support the burst bus cycles.

2.11.2.1 Burst Breakpoint

Burst instruction breakpoints do not terminate the burst bus cycle. Individual words in a burst cycle can be breakpointed. The instruction breakpoint is taken when the word breakpointed is accessed in the pipe. If the pipe and queue are flushed by a change of flow the breakpoint will not be taken.

2.11.2.2 Burst Bus Error

Bus error terminates the burst bus cycle in progress. The bus error tags the last word transferred during the burst in the instruction queue with bus error. When the word with the bus error tag reaches the pipe the bus error exception is taken. Until the bus error exception is taken more burst prefetches and operand accesses can occur. If the instruction bus error is flushed from the pipe and queue by a branch instruction or other change of flow the bus error will not be taken.

Late bus errors on burst bus cycles are not supported.

2.12 Interrupt Response Time

The minimum interrupt response time of the CPU32X is improved over the CPU32. The CPU32 requires a minimum of thirty clocks to begin executing the first instruction of the interrupt service routine and the CPU32X will require a minimum of eighteen clocks or twelve clocks less. Table 2-5 indicates the minimum number of clocks required and the operations performed for the CPU32 and the CPU32X to begin the interrupt service routine. This example assumes that the stack **and vector table** are in the FASRAM, external memory accesses are two cycle accesses and burst memory is a **1,1,1,1 external** burst memory, which is a **2,1,1,1 internal** burst memory. Additional clocks for slower external burst devices and non-FASRAM vector table accesses will increase the interrupt response time.

Table 2-5 Interrupt Response

Clock	CPU32X	CPU32
	Last instruction	Last instruction
1		
2	IACK cycle	
3		IACK cycle
4		
5		
6		
7		
8	Stack vector	
9		
10	Stack SR	
11	Stack PCH	Stack vector
12	Stack PCL	
13	**Read vector high**	
14	**Read vector low**	
15	Begin burst	Stack SR
16	I0	
17	I1	Stack PCH
18	I2	
19	**Begin Interrupt Routine**	Stack PCL
20		
21		Read vector high
22		
23		Read vector low
24		
25		Fetch first instruction
26		
27		Fetch second instruction
28		
29		Fetch third instruction
30		
		Begin interrupt routine

2.13 Power

The power dissipation of the greater-than-two-times faster CPU32X is substantially less than two times the CPU32 power dissipation. This power reduction is from improving the power dissipation in the CPU32 core, by stopping all unnecessary clocks and by design for power in the new burst mode, fast bus and instruction queue logic. In addition the operation of the stop instruction will be improved to reduce power when the CPU32X is stopped.

2.14 Debugging Support

The CPU32X provides features that facilitate applications development similar to the CPU32 debugging features. These features include background debug mode, breakpoint, and pipe tracking.

2.14.1 Background Debug Mode

This feature remains the same on CPU32X as on CPU32.

2.14.2 Breakpoint

This feature is enhanced from the CPU32 to provide burst breakpoints.

2.14.3 Pipe Tracking

CPU32X uses three pins. IPIPE[2:0] provides five types of information for the bus analyzer to track the pipe. $\overline{\text{IPIPE[0]}}$ and $\overline{\text{IPIPE[1]}}$ both carry active low, time multiplexed signals, and provide four types of information, which are instruction-start, opcode-advance, instruction-fetch, and pipe-flush. IPIPE[2] indicates the number of instructions in the pipe, and helps the bus analyzer synchronize with the pipe.

2.14.3.1 Instruction Queue and Pipeline

In addition to the original three-stage instruction pipeline, CPU32X implements a six-word instruction queue. The internal instruction queue and pipeline can be modeled as a nine-stage FIFO and a fetch pointer. **Table 2-6** describes the FIFO.

IR contains the currently executing instruction. IR0 contains instruction extension words. When a new opcode or extension word is used, the cpu will shift the data in the instruction registers upward, and discard the old data in IR or IR0.

The fetch pointer contains the address of the empty instruction register to be filled by the new incoming instruction word. The fetch pointer may point to IR0~IR7, but it will never directly point to IR.

Table 2-6 Nine-Stage Instruction Registers

Address	Instruction Register	Data	
	IR (IRC)	Current instruction	
000	IR0(IRB)	Extension word	
001	IR1(IRA)	New opcode	
010	IR2	New opcode	
011	IR3	New opcode	
100	IR4	Empty	← Fetch Pointer
101	IR5	Empty	
110	IR6	Empty	
111	IR7	Empty	

2.14.4 PIPE Tracking Operations

2.14.4.1 $\overline{\text{IPIPE[0]}}$

The $\overline{\text{IPIPE[0]}}$ pin provides the instruction-start and opcode-advance signals. The start signal is derived from sampling $\overline{\text{IPIPE[0]}}$ at the falling edge of the system clock, and the advance signal is derived from sampling $\overline{\text{IPIPE[0]}}$ at the rising edge of the system clock. They are both active low signals.

Instruction-start and opcode-advance both indicate the use of the new instruction word in the pipeline. Instruction-start means the start of a new instruction. The old opcode in IR is replaced by the new opcode in IR0, and all of the data in the pipeline is shifted upward, (i.e., IR0->IR, IR1->IR0, ... and IR7->IR6). The fetch pointer is decremented by one.

Opcode-advance is different from instruction-start. It indicates the use of extension word from IR0. The data in IR0 is replaced by IR1 and all of the data is shifted upward to IR0, (i.e., IR1->IR0, IR2->IR1,..., and IR7->IR6). The fetch pointer is also decremented by one.

2.14.4.2 $\overline{\text{IPIPE[1]}}$

The $\overline{\text{IPIPE[1]}}$ pin provides pipe-flush and instruction-fetch information. The flush signal is derived from sampling $\overline{\text{IPIPE[1]}}$ at the rising edge of system clock, and the fetch signal is derived from sampling $\overline{\text{IPIPE[1]}}$ at the falling edge of system clock. The fetch and flush signals are both active low.

Pipe-flush indicates when a change of flow occurs and all of the instructions in the pipeline are flushed. After pipe-flush is asserted, the pipeline is empty, and the fetch pointer is zero.

Instruction-fetch indicates that the data from the current IMB cycle is to be routed to the instruction register pointed to by the fetch pointer. After the data is copied to the instruction register, the fetch pointer is incremented by one. The fetch pointer now points to the next empty instruction register.

The instruction-fetch signal is asserted at IMB B3 state, and repeats asserted during wait states until B4.

Figure 2-3 describes the timing of $\overline{\text{IPIPE[1:0]}}$ pins and the fetch pointer.

Figure 2-3 through **Figure 2-8** describe the timing of the IPIPE[1] in different burst fetch cycle. **Figure 2-5** shows minimum logic required to demultiplex $\overline{\text{IPIPE[0]}}$ and $\overline{\text{IPIPE[1]}}$.

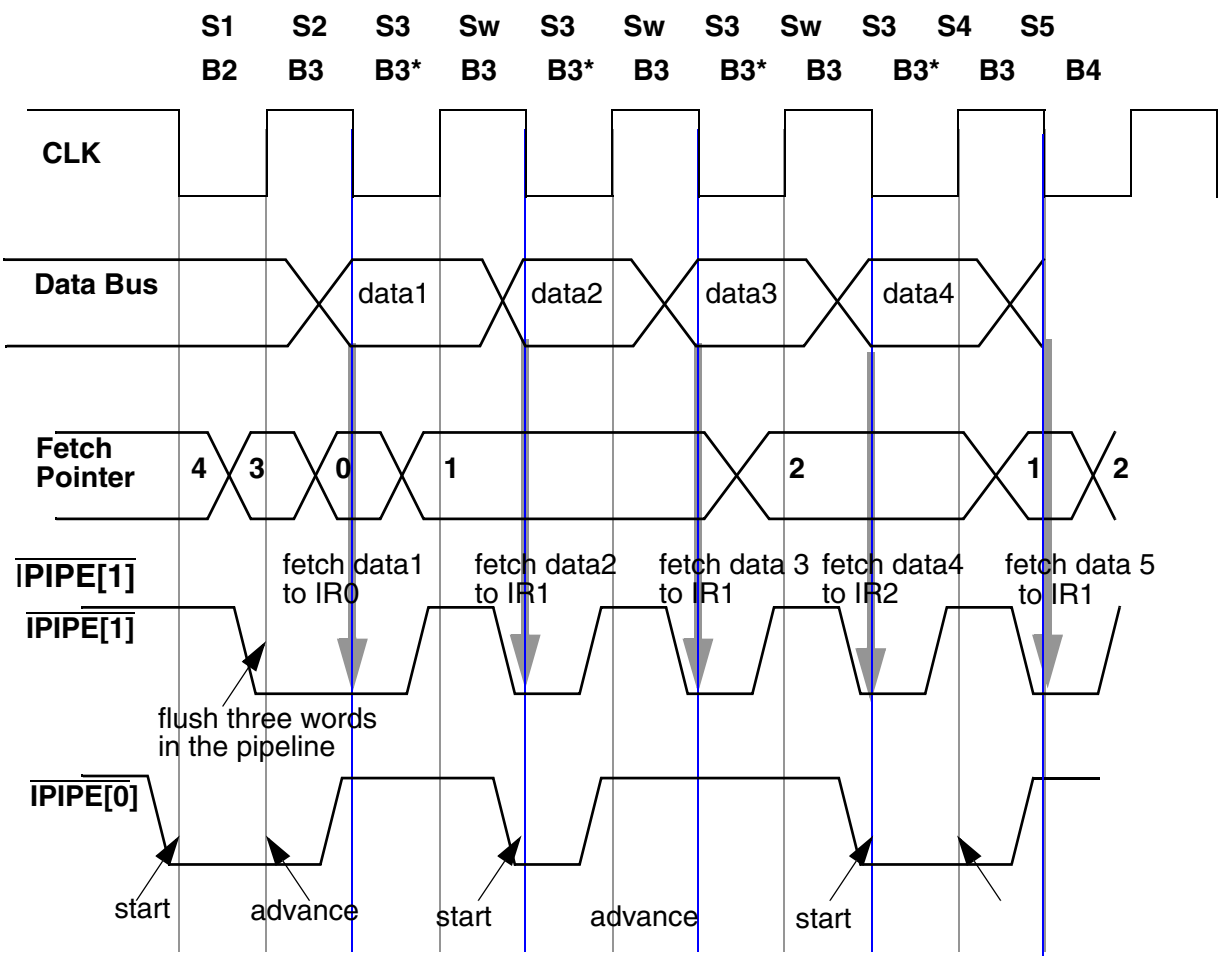


Figure 2-3 $\overline{\text{IPIPE[1:0]}}$ Timing Diagram

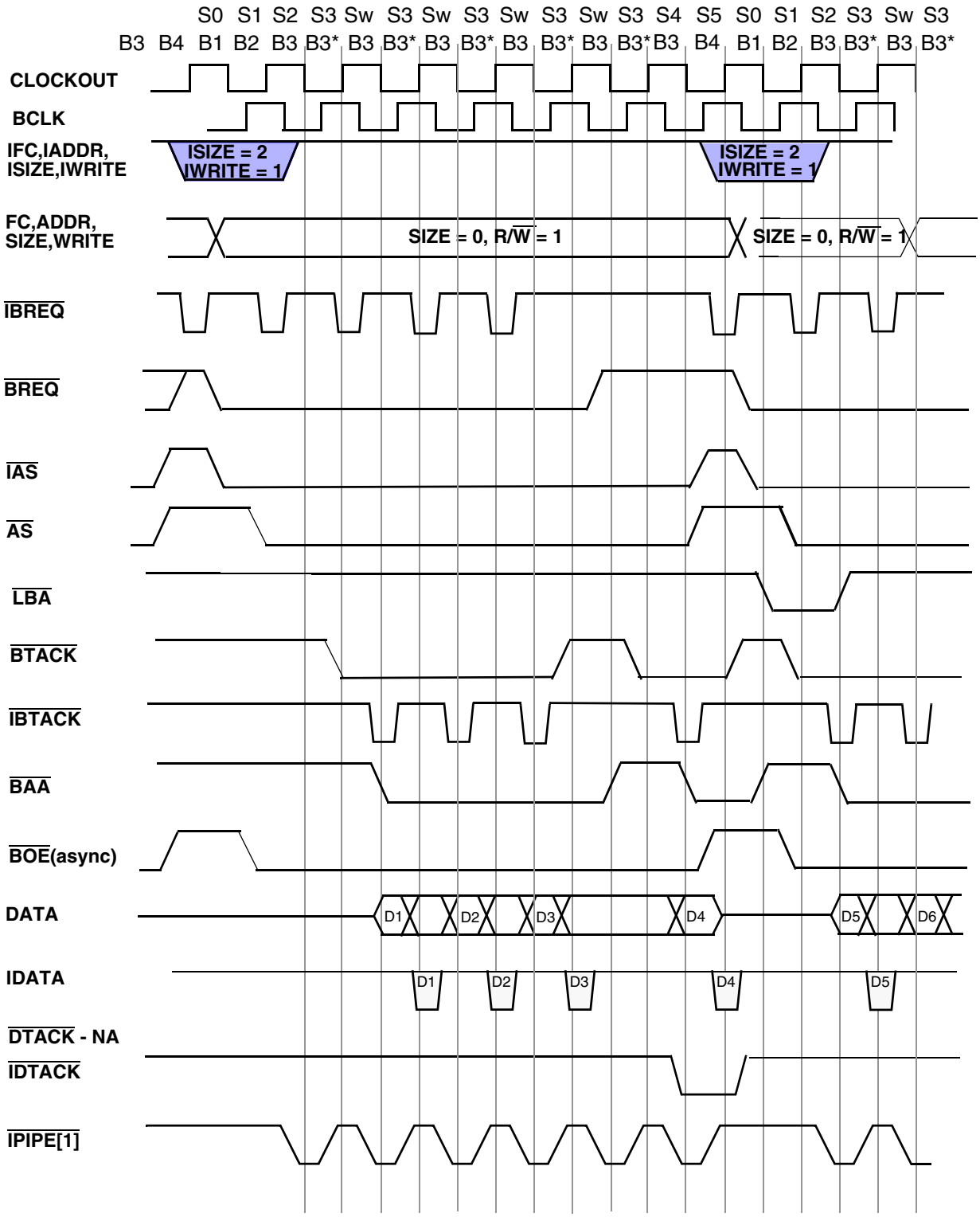
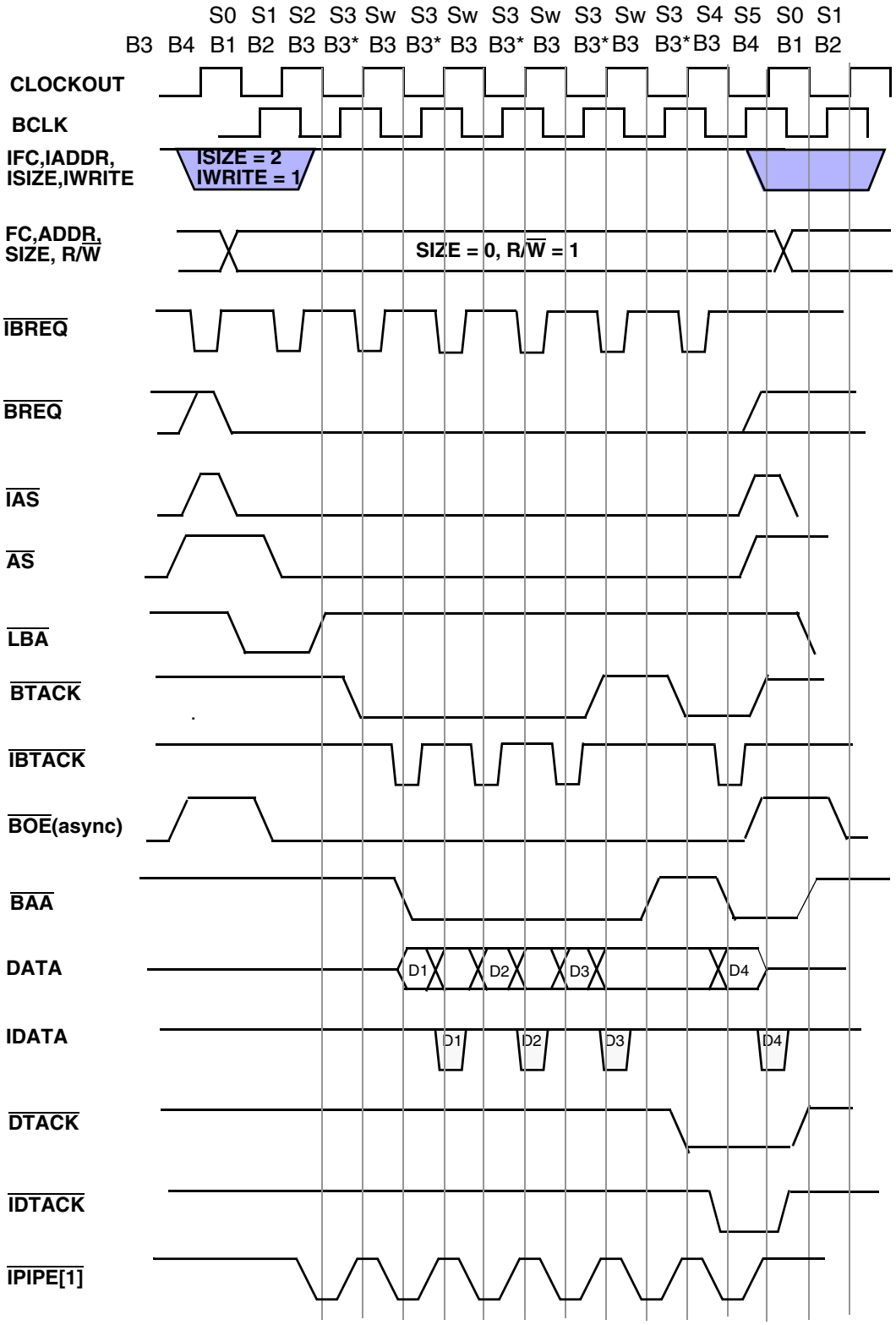


Figure 2-4 IPIPE[1] Timing Diagram In a Burst Fetch Cycle (Part 1)



NOTE: Slave terminates cycle by asserting DTACK.
IPIPE[1] indicates the data, D1~D4, enters pipeline.

IPIPE[1:0] Timing Diagram in a Burst Fetch Cycle (Part 2)

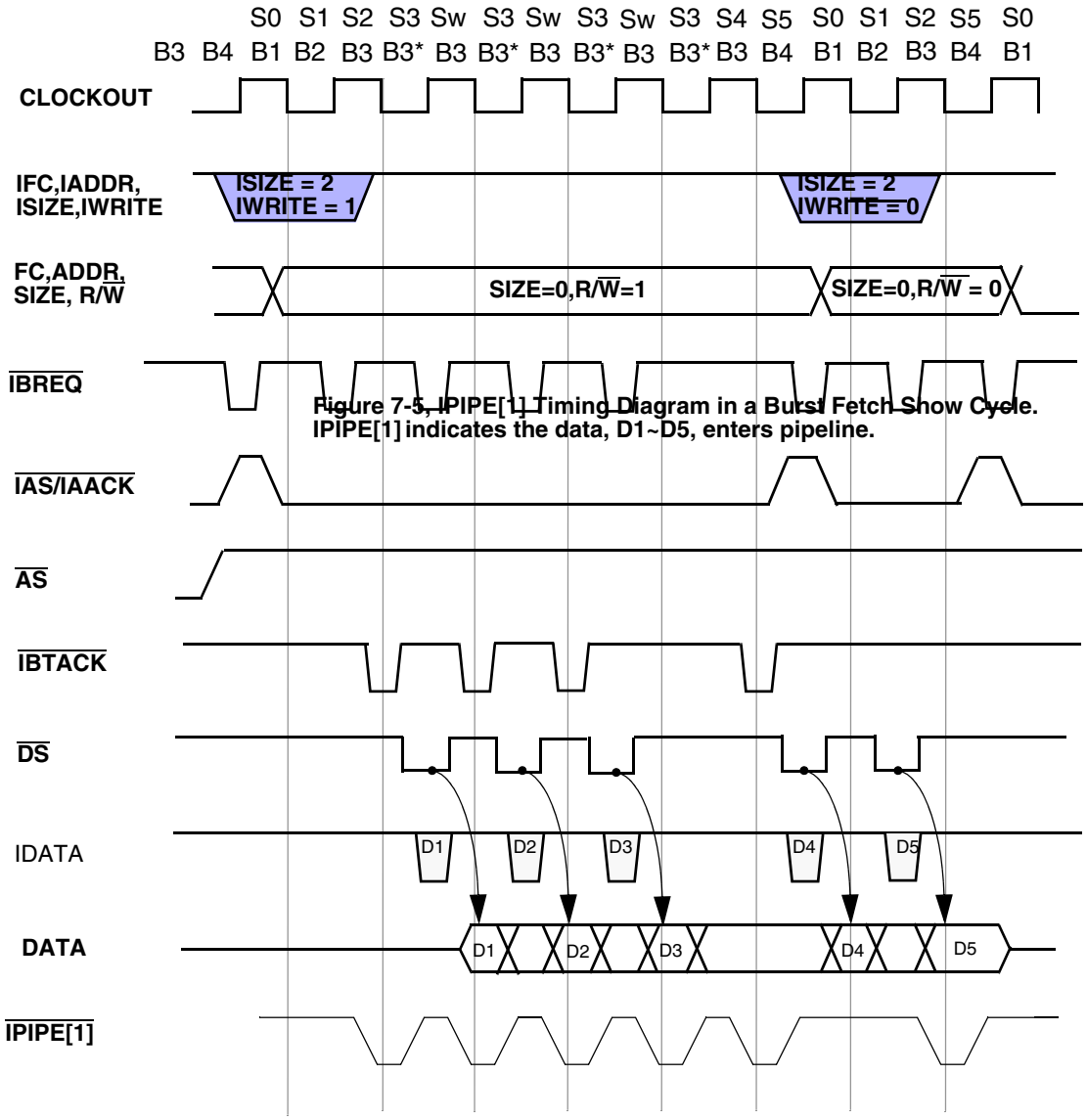


Figure 7-5, IPIPE[1] Timing Diagram in a Burst Fetch Show Cycle. IPIPE[1] indicates the data, D1~D5, enters pipeline.

IPIPE[1:0] Timing Diagram in a Burst Fetch Cycle (Part 3)

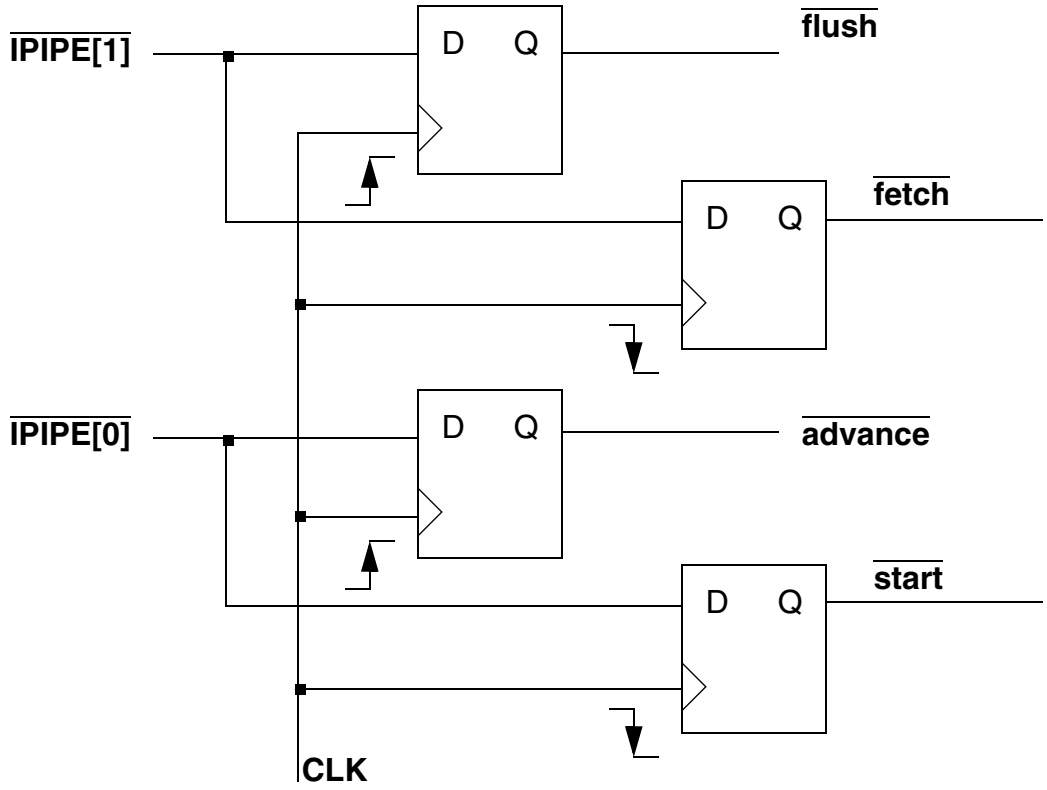


Figure 2-5 IPIPE[1:0] DEMUX Logic

2.14.4.3 IPIPE[2]

In the CPU32, the only way to synchronize the bus analyzer with the internal pipe at the first start is through a pipe_flush signal. Once the bus analyzer receives the signal, it knows the pipeline is empty and then starts the pipe tracking. However, since the pipe_flush signal depends on the change-of-flow signal from the user's program, there is a chance that the bus analyzer has to wait for a long time before the pipe_flush is generated and can not service the user at all.

In the CPU32X, IPIPE[2] pin provides the pipe synchronization function to help the bus analyzer synchronize with the internal pipe without depending on the pipe_flush signal. A series of bits (see Figure 2-3) are sent out by IPIPE[2] to indicate the number of instruction words in the pipeline. Once the bus analyzer gets the information, it can assign the fetch pointer to the correct address and start tracking the pipe from that time point.

The serial data bits start from one start bit ("0"), followed by three bits, indicating the number of instruction words in the pipeline, and then followed by three or more stop bits ("111"). The start bit always comes with a fetch or a flush signal, called fetch or flush. The IPIPE[2] data indicates the number of instructions in the pipeline after the fetch or flush occur. When it comes with flush, the data from the next fetch cycle must

enter IR0. When it comes with fetch, the instruction number derived from IPIPE[2] is assigned as the fetch pointer for the fetch cycle following fetch.

If there are start or advance signals detected between fetch and the next fetch, the fetch pointer has to be adjusted by subtracting the number of starts and advances to acquire the correct fetch pointer.

In the example of **Figure 2-3,** the fetch pointer is "011" after fetch1 occurs. However, a start occurs simultaneously with the following fetch cycle, so the data2 from the following fetch enters IR2 again. The fetch pointer is "101" after fetch2 occurs and therefore, data7 from the following fetch enters IR5.

Figure 2-7 lists the detailed timing for IPIPE[2:0] and shows an example of how to use IPIPE[2] to synchronize the pipe tracking function in tabular form.

In **Figure 2-7**, the first IPIPE[2] start bit is detected at line one, which comes with a flush signal; therefore, the data of the following fetch cycle enters IR0. The second IPIPE[2] start bit is detected at line 17, and the IPIPE[2] data indicates there are three instruction words in the pipeline before the next fetch (at line 19) occurs. The fetch pointer is set to '3' at line 18. However, there is another "instruction-start" detected at line19, so the fetch pointer for the fetch cycle at line19 is adjusted to two by subtracting '1'.

The third IPIPE[2] start bit is detected at line 33, which comes with a fetch cycle at line 33. The IPIPE[2] data indicates the following incoming data should enter IR0. In this case, it means the pipeline is full and if another fetch cycle occurs before any "instruction-start" or "instruction-advance", the pipe will overflow, which is impossible to happen. At line 34, the fetch pointer is assigned to '0' according to the data from IPIPE[2]. But before the next fetch cycle comes, an "instruction-start" and "instruction-advance" are detected, and the fetch pointer should be adjusted by adding '8' and subtracting '2' for the fetch cycle at line 39.

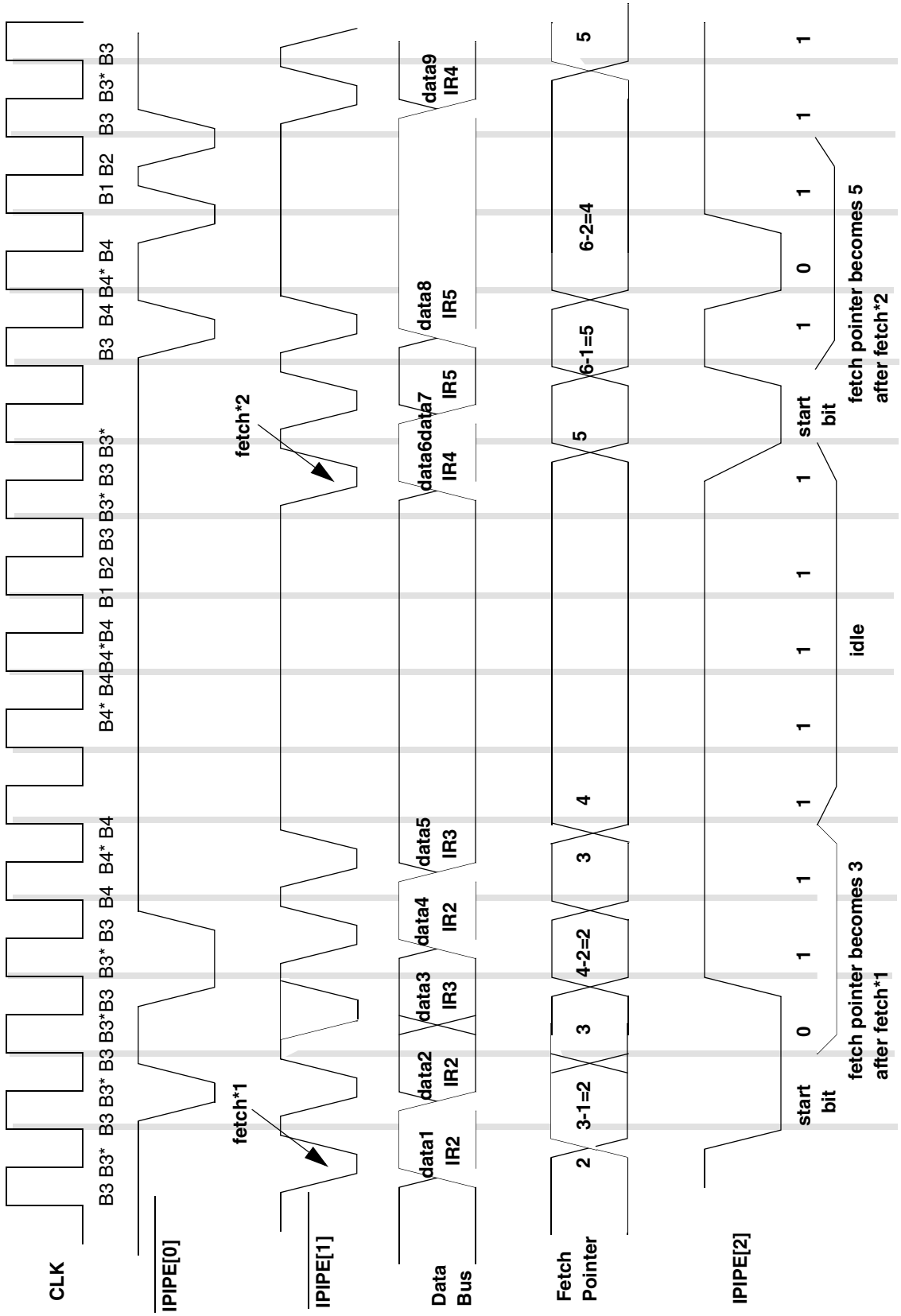


Figure 2-6 $\overline{\text{IPIPE[2]}}$ Timing Diagram

Table 2-7 How to Use $\overline{\text{IPIPE[2]}}$ to Synchronize Pipe Tracking

line	clkout	ipipe2	flush	advance	fetch	start	latch data	data ->IR?	fetch pointer
1	^	0	1	1	x	1	xxxx		
2		0	1	1	x	1	xxxx		0
3	^	0	0	1	x	1	xxxx		0
4		0	0	1	0	1	2a7c		0
5	^	0	1	1	0(+)	1	2a7c	IR0	0
6		0	1	1	0	1	0020		1
7	^	0	1	1	0(+)	1	0020	IR1	1
8		0	1	1	0	1	4a00		2
9	^	1	1	1	0(+)	1	4a00	IR2	2
10		1	1	1	0	0	4e68		3
11	^	1	1	0(-)	0(+)	0(-)	4e68	IR1	1
12		1	1	0	0	1	4e60		2
13	^	1	1	0(-)	0(+)	1	4e60	IR1	1
14		1	1	0	0	1	4e7a		2
15	^	1	1	1	0(+)	1	4e7a	IR2	2
16		1	1	1	0	0	6801		3
17	^	0	1	1	0(+)	0(-)	6801	IR2	2
18		0	1	1	0	0	48d5		3
19	^	0	1	1	0(+)	0(-)	48d5	IR2	2
20		0	1	1	0	0	5555		3
21	^	1	1	0(-)	0(+)	0(-)	5555	IR1	1
22		1	1	0	0	1	0f8e		2
23	^	1	1	1	0(+)	1	0f8e	IR2	2
24		1	1	1	0	1	0000		3
25	^	1	1	1	0(+)	1	0000	IR3	3
26		1	1	1	0	1	700a		4
27	^	1	1	1	0(+)	1	700a	IR4	4
28		1	1	1	0	1	0e15		5
29	^	1	1	1	0(+)	1	0e15	IR5	5
30		1	1	1	0	1	1800		6
31	^	1	1	1	0(+)	1	1800	IR6	6
32		1	1	1	0	1	c7c2		7
33	^	0	1	1	0(+)	1	c7c2	IR7	7
34		0	1	1	1	0			0
35	^	0	1	0(-)	1	0(-)			0-2+8
36		0	1	0	1	1			6
37	^	0	1	1	1	1			6
38		0	1	1	0	1	xxxx		6
39	^	0	1	1	0(+)	1	xxxx	IR6	6
40		0	1	1	1	1			7

2.14.4.4 Loop Mode

$\overline{\text{IPIPE}}[1]$ and IPIPE[2] continue to work normally during loop mode. However, unlike CPU32, CPU32X does not discard any opcode since the opcodes are being reused during loop mode. The start and advance signals will not be asserted during loop mode. From the user's point of view, loop mode is like running a long instruction. Figure 2-8 is an example of loop mode.

Assembly Code

```
203c  0000 0000          move.l #$0,d0
207c  0000 1000          move.l #$1000,a0
d441                     add d1,d2
30c0                lp   move d0,(a0)+
54ca  fffc               dbcc d2,lp
d642                     add d2,d3
d84b                     add d3,d4
                          .
                          .
                          .
```

sync	$\overline{\text{fetch}}$		$\overline{\text{advance}}$						fetch	operand	
	$\overline{\text{flush}}$		$\overline{\text{start}}$		IR	IR0	IR1	IR2	opcode	addr.	data
1	1	1	1	1	d441	30c0	54ca	fffc			
1	1	1	0	1	30c0	54ca	fffc	d642			
1	1	1	1	1	30c0	54ca	fffc	d642		1000	0000
1	1	1	0	1	54ca	fffc	d642	d84b			
0	1	1	1	0	54ca	d642	d84b				
0	0*	1	1	1	xxxx	xxxx	xxxx	xxxx			
0	1	1	1	1	xxxx	xxxx	xxxx	xxxx			
0	1	0	1	1	xxxx	30c0	xxxx	xxxx	30c0		
1	1	0	0	1	30c0	54ca	xxxx	xxxx	54ca		
1	1	0	1	1	30c0	54ca	fffc	xxxx	fffc	1002	0000
1	1	0	0	1	54ca	fffc	d642	xxxx	d642		
1	1	0	1	1	54ca	fffc	d642	d84b	d84b	1004	0000
1	1	0	1	1	54ca	fffc	d642	d84b			
1	1	0	1	1	54ca	fffc	d642	d84b		1006	0000
1	1	1	1	1	54ca	fffc	d642	d84b			
1	1	1	1	1	54ca	fffc	d642	d84b		1008	0000
1	1	1	1	1	54ca	fffc	d642	d84b			
1	1	1	1	1	54ca	fffc	d642	d84b		100a	0000
1	1	1	1	1	54ca	fffc	d642	d84b			
1	1	1	1	0	54ca	d642	d84b				
1	1	1	0	1	d642	d84b					

Loop Mode

Figure 2-7 Pipe Tracking Operation In Loop Mode

2.14.5 Pipe Tracking Flowchart

Figure 2-9 is the pipe tracking flowchart. The bus analyzer should follow it to track the pipeline.

Master terminates cycle by negating IBREQ. IPIPE[1] indicates the data, D1-D6, enters pipeline.

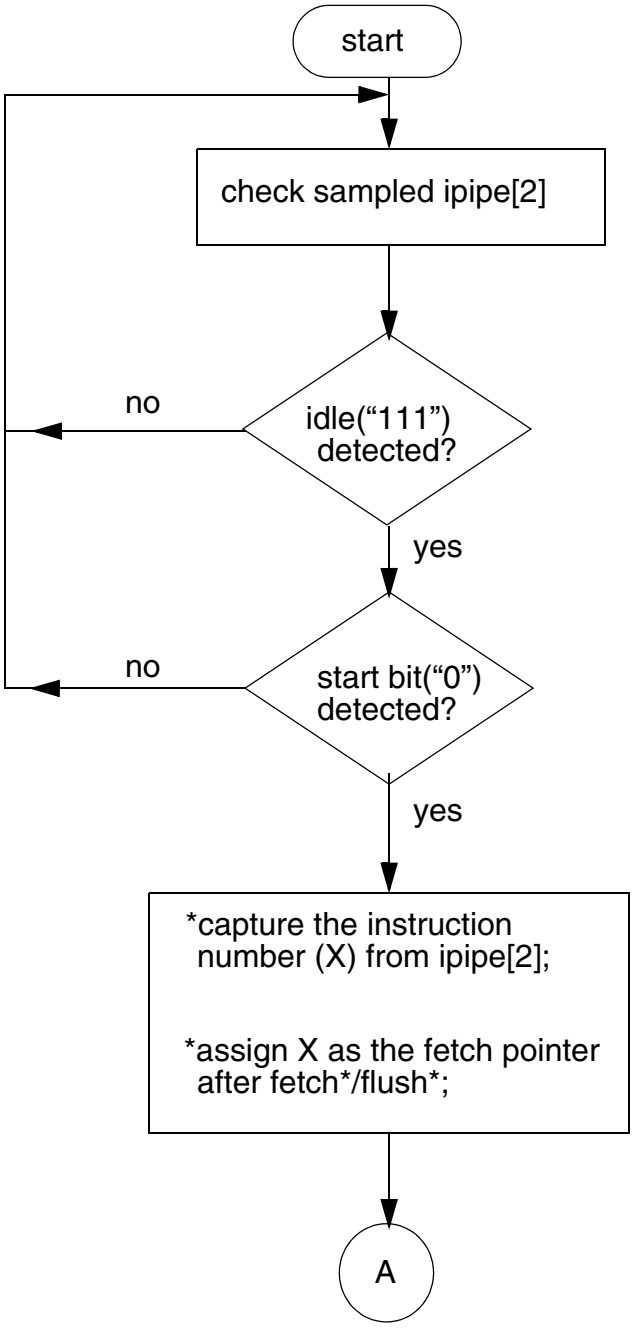


Figure 2-8 Pipe Tracking Flowchart (Part 1)

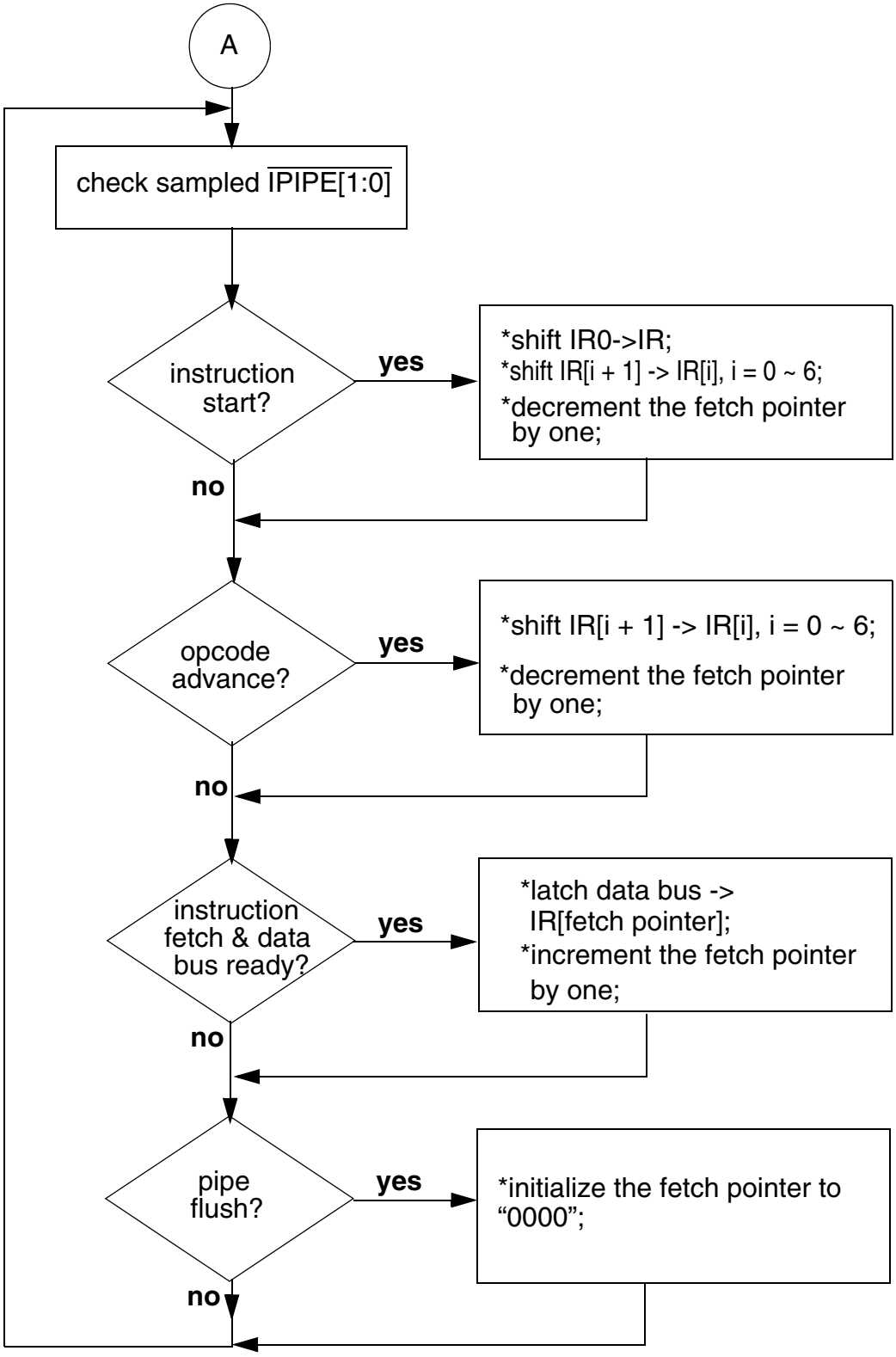


Figure 2-9 Pipe Tracking Flowchart (Part 2)

2.15 IMB Interface and External Pins

All of the IMB pins currently supported by the CPU32 will be supported by the CPU32X. The IMB burst protocol pins, BTACK and BREQ are added to the IMB interface. One pin will be added, in addition to the IPIPE and IFETCH pins, to support pipe and instruction queue tracking.

2.16 Basic Electrical Specifications

Table 2-8 Electrical Specifications

Operating Range	Temperature Range in C	Min/Max VDD in V	Max Fsys in MHz	Max Current in mA
Automotive	-40 to 125	4.9 to 5.1	21.75	90
Commercial	-40 to 110	4.75 to 5.25	25	105
Low voltage commercial	-40 to 85	2.7 to 3.3	8	20

2.17 Test Features

The test features of the CPU32X will include at least the test features of the CPU32. Any additional test features are to be determined.

SECTION 3
BURST INTEGRATION MODULE (BIM)

3.1 Introduction

3.1.1 Features

The burst integration module (BIM) provides two-fold improvement in instruction bandwidth, as compared to previous integration modules, by utilizing a synchronous burst protocol. BIM applications also realize decreased memory costs due to a nearly one-half clock speed improvement in memory access timing.

The BIM has a very low number of pins which are required for basic operation and test. It has the ability to support additional incremental features if pins are available, without the need to redesign the BIM. The pins required to support the incremental features are shown in **Figure 3-1** and denoted by an asterisk (*). Incremental features are configured by external (from the module) connections to optional pads, or jumpers if pads are not available for the function.

The BIM also has the capability to control dual voltage drivers. When a pin is configured as its primary function, then the BIM will enable a 3-V driver in the associated pad. When a pin is configured as digital I/O, then the BIM will enable a 5-V driver with slow rise time in the associated pad to minimize electromagnetic interference (EMI) in the application.

3.1.2 Sub Modules

Figure 3-1 is a BIM module block diagram.

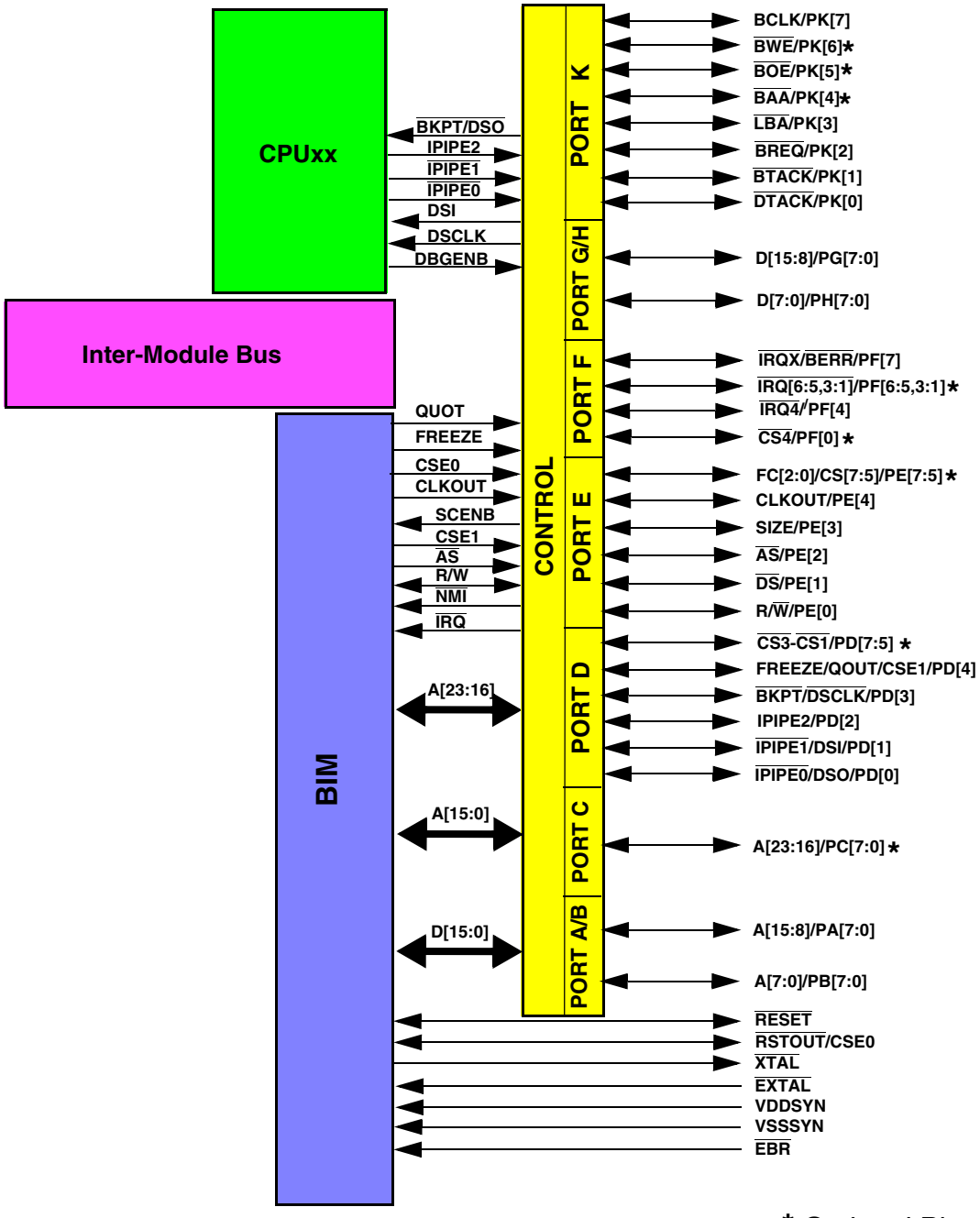


Figure 3-1 BIM Block Diagram

3.1.3 Clocks

The BIM provides the system clocks for the device on which it is implemented, as well as an external clock signal (CLKOUT). The system clocks are generated from the internal phase locked loop (PLL) at a frequency which is derived from a reference fre-

quency. The reference frequency source can be either the crystal oscillator reference or an external clock reference. The BIM provides the user with several options to control the clock frequency of the host device and external clocks, both in normal operation and standby modes. A complete description of the clock circuit is included in **3.4 Clocks**.

3.1.4 BIM Operation

The BIM can operate in master mode, single chip mode, or emulation mode. The BIM's operating mode is determined at reset and thereafter cannot be changed.

Master mode operation allows the internal CPU to access both internal and external memories and peripherals. The external bus consists of a 16-bit data bus, 16-24 address lines, and optional function code lines. Available bus control signals include $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$, SIZE, $\overline{BREQ}$, $\overline{BTACK}$, $\overline{BERR}$ and $\overline{DTACK}$. With the burst chip select (BCS) interface, an external burst memory can be selected to increase the performance of the system. The asynchronous chip selects (ACS) can be programmed to select and control external devices, and provide bus cycle termination. Seven interrupt levels are supported with up to seven external interrupt pins. A number of the external pins can be configured for digital I/O. This configuration is shown in **Figure 3-2**.

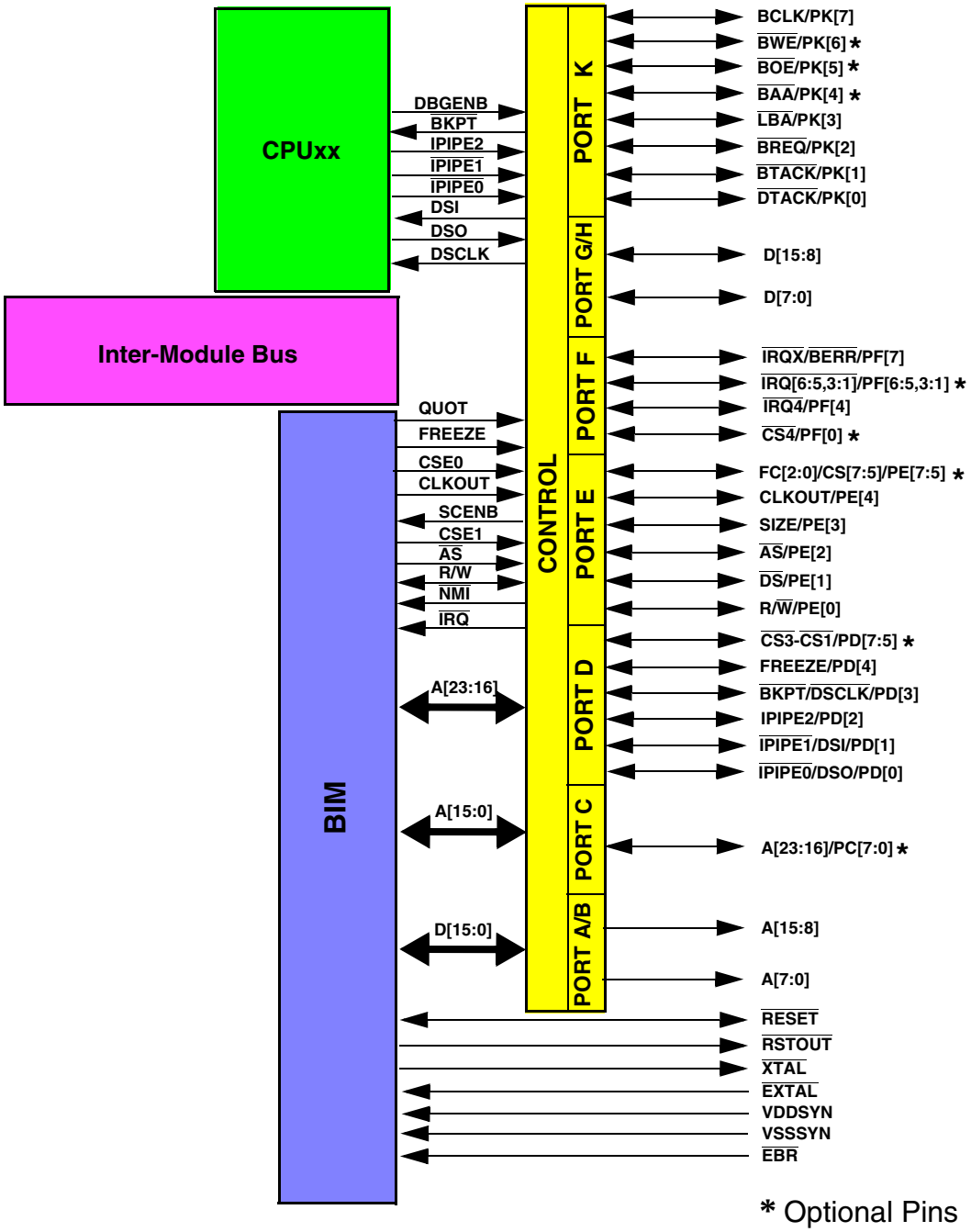


Figure 3-2 BIM Master Mode Configuration

In single chip mode, the CPU and system memory are on chip. External bus pins are either configured as specified by the PCON options or as digital I/O as specified in

Table 3-5.Seven interrupt levels are supported with up to seven external interrupt pins. CLKOUT can be used to provide a system clock. This configuration is shown in Figure 3-3.

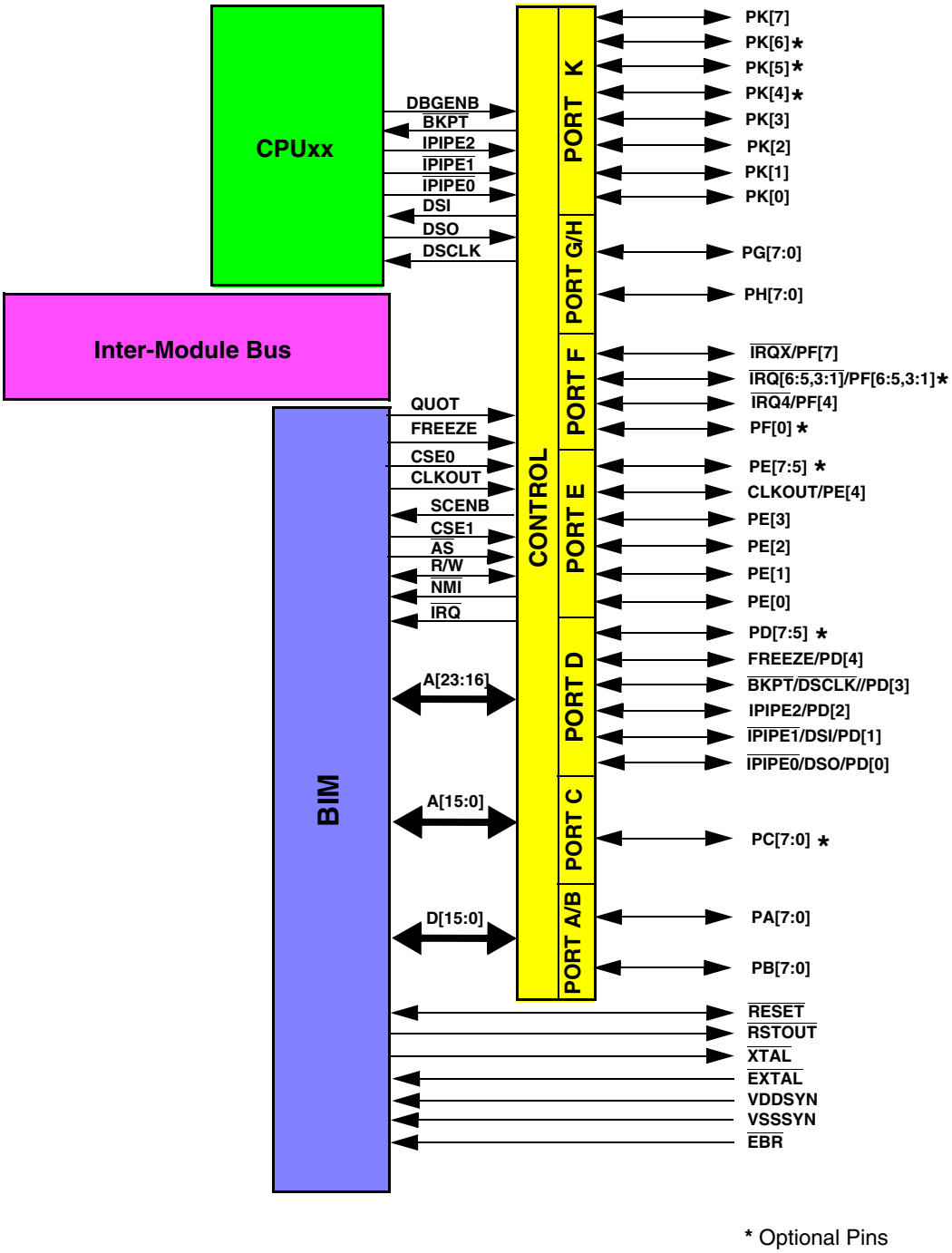
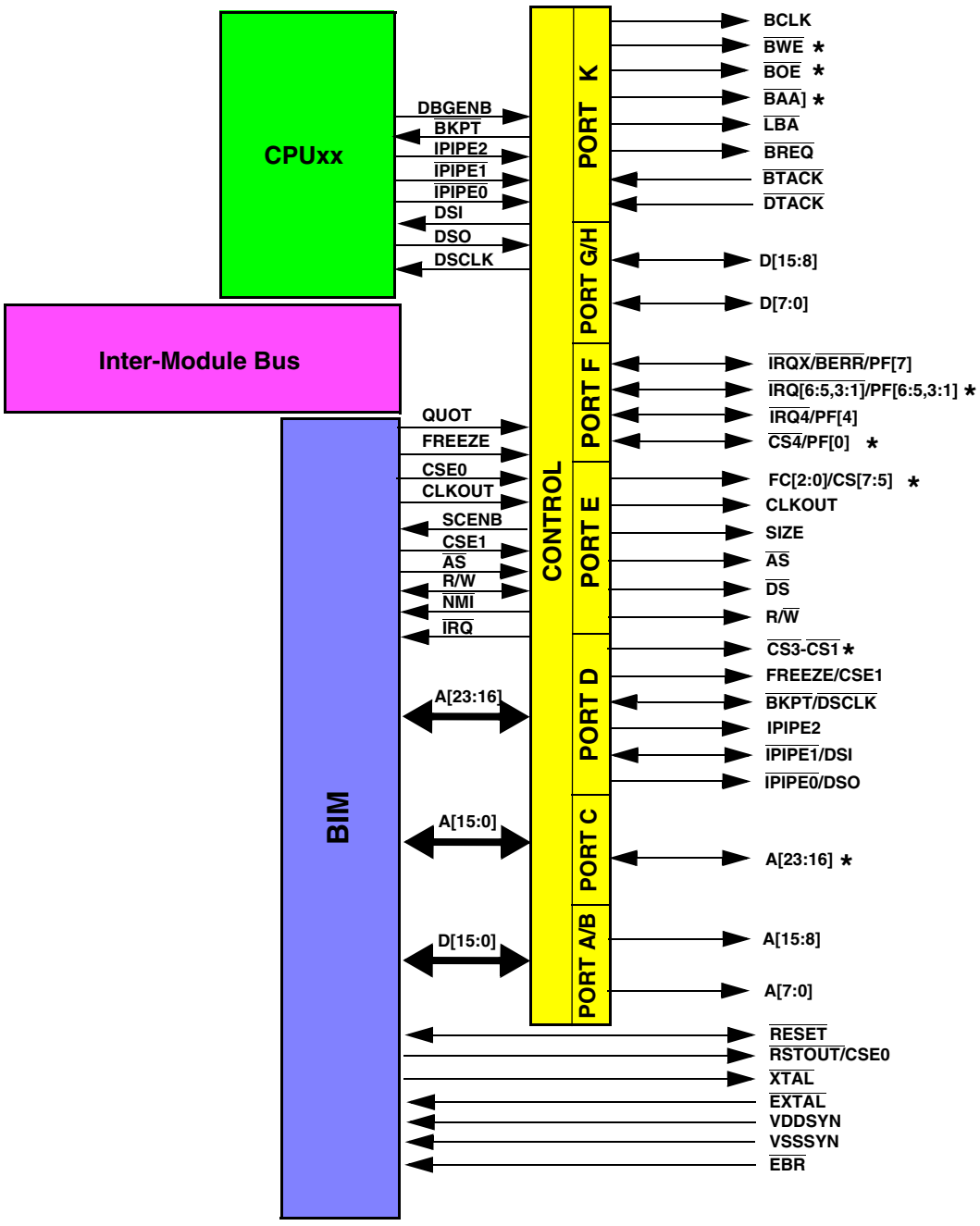


Figure 3-3 BIM Single Chip Configuration

Emulator mode is designed to improve the visibility of the overall system, in applications which utilize digital I/O pins. In emulator mode, the MCU, external memory and port replacement logic are used in combination to mimic the final system application. Since the MCU's external pins are not available to perform their digital I/O function in emulator mode, all port functions (except port F) need to be provided by the port replacement logic.

In emulator mode, primary pin functions are enabled (per PCON), except port F which can be either interrupts or digital I/O. Since the full external bus must be visible to support the external memory and port replacement logic, the emulation mode pin configuration resembles master mode. The external bus consists of a 16-bit data bus, 16-24 address lines, and optional function code lines. Available bus control signals include $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$, SIZE, $\overline{BREQ}$, $\overline{BTACK}$, $\overline{BERR}$ and $\overline{DTACK}$. The burst chip select interface can be used with an external burst memory to emulate an internal burst memory. The asynchronous chip selects can be programmed to select and control external devices, and to provide bus cycle termination. Also emulation mode chip selects, CSE1 and CSE0, are provided to give additional information about the bus cycle.This configuration is shown in **Figure 3-4**.



* Optional Pins - not required for minimum BIM implementation

Figure 3-4 BIM Emulator Mode Configuration

3.1.5 RESET Configuration

The BIM has several sources of reset which include double bus fault monitor, software watchdog reset, power-on reset, loss of crystal reset, test module reset, and external reset. For more information, refer to **3.4.8 RESET Operation**.

During reset, the state of data pins D[15:12, 10] are driven onto IMB data lines ID[15:12, 10], to provide internal module configuration information. The state of data bus pins (D[11, 10:0]), $\overline{\text{SIZE}}$, $\overline{\text{LBA}}$, $\overline{\text{BTACK}}$, $\overline{\text{BOE}}$, $\overline{\text{BREQ}}$ and $\overline{\text{DTACK}}$ are used to configure various functions in the pin assignment registers. During reset these pins have [internal] weak pull-up or pull-down devices controlled by the PCON/override characteristics.

NOTE

Special care must be taken to ensure that the application uses these override pins as OUTPUTS. Futhermore, these pins should not have any devices connected to them that would pull these signals high or low during RESET unless the application truly intends to override the PCON value. If they are used in the application as inputs, during RESET the state of application signals may erronously override the PCON value causing undefinded behavior.

A[23:0], $\overline{\text{IPIPE0}}$, $\overline{\text{IPIPE1}}$, $\overline{\text{IPIPE2}}$, $\overline{\text{AS}}$, $\overline{\text{DS}}$, R/$\overline{\text{W}}$, $\overline{\text{CS1}}$ – $\overline{\text{CS7}}$, FREEZE, $\overline{\text{IRQX}}$, $\overline{\text{IRQ}}$[6:1], $\overline{\text{BAA}}$, and $\overline{\text{BWE}}$ are all tristated in reset (internal pull-ups will pull the pins high).

To be compatible with the SLIM operating in peripheral mode, A[15:0] will tristate during reset (internal pull-ups will pull the pins high).

The $\overline{\text{RSTOUT}}$ pin is driven low while the RESET pin is asserted, to indicate that the internal reset controller has reset the chip.

All pins which are configured as digital I/O during reset will be configured as inputs when the BIM comes out of reset.

NOTE

Port F unimplemented pins must be reassigned to digital I/O after reset to avoid spurious interrupts.

For more information on pin assignments at reset, refer to **3.2 BIM Ports**.

3.1.5.1 Operation Mode Selection

The BIM's operating mode is selected during reset and recorded in bits [11:9] of the BIM module configuration register (MCR). The encodings are shown in **Table 3-1**. Once reset is exited, the operating mode cannot be changed. The operating mode selection process is tailored for customer applications, but overrides are defined to allow any mode to be specified.

To simplify the customer application, bits [11:10] of the MCR are initialized to ones and a customer-specified, mask-programmable bit in the MCR is provided to allow the BIM

to be automatically configured for either master mode or single chip mode. If this default operating mode is not desired, the mode can be overridden at reset by driving D[0], to select the alternate application operating mode.

Table 3-1 Operation Mode Encoding

Operation Mode	MCR[11:9]
Master mode	1 1 1
Single chip mode	1 1 0
Slave access mode	1 0 0
Emulation mode	0 X X

Table 3-2 Operation Mode Selection

Operation Mode	BIT 11	BIT 10	BIT 9
Master mode or Single chip mode	1 (default)	1 (default)	D[0] may be driven to override mask-programmable value
Emulation mode	Set to 0 by D[10]	Don't care	Don't care

3.1.5.2 Pin Function Selection

Most BIM pins support at least two functions. Immediately after reset (internally or from external logic), the BIM pins are configured based upon the operating mode, PCON values, and whether background mode is enabled. These default pin functions are shown in **Table 3-3**.

NOTE

In some cases the default pin function is not fixed, but is selected by customer-specified, mask-programmable bits in the port configuration register (PCON). These mask-programmable defaults can be overridden at reset by driving external pins.

In most cases, the pin function can be changed after reset by writing to the appropriate pin assignment register (PAR). For specific information see **3.2 BIM Ports**.

Table 3-3 Default Pin Function Immediately after RESET

Signal	Port	RESET Pin Function in Each Mode (Background Enabled)			
		Master	Single Chip	SLAM	Emulation
A[15:0]	A,B	A[15:0]	PA[7:0], PB[7:0]	A[15:0] (input)	A[15:0]
		A[15:0]	PA[7:0], PB[7:0]	NA	A[15:0]
A[23]	C[7]	PCON[9][1]	PCON[9][1]	PCON[9][1]	PCON[9][1]
		PCON[9][1]	PCON[9][1]	NA	PCON[9][1]
A[22:16] (optional)	C[6:0]	PCON[13:11][1]	PCON[13:11][1]	PCON[13:11][1]	PCON[13:11][1]
		PCON[13:11][1]	PCON[13:11][1]	NA	PCON[13:11][1]
CS3 – CS1 (optional)	D[7:5]	PCON[5][1]	PCON[5][1]	PCON[5][1]	PCON[5][1]
		PCON[5][1]	PCON[5][1]	NA	PCON[5][1]
FREEZE / QUOT / CSE1	D[4]	PD[4]	PD[4]	FREEZE (input)	FREEZE /CSE1
		FREEZE	FREEZE	NA	FREEZE /CSE1
$\overline{\text{BKPT}}$ / DSCLK	D[3]	PD[3]	PD[3]	PD[3]	BKPT
		BKPT	BKPT	NA	BKPT / DSCLK
IPIPE2	D[2]	PD[2]	PD[2]	PD[2]	IPIPE2
		IPIPE2	IPIPE2	NA	IPIPE2
$\overline{\text{IPIPE1}}$ / DSI	D[1]	PD[1]	PD[1]	$\overline{\text{LATCH}}$ (input)	IPIPE1
		IPIPE1	IPIPE1	NA	IPIPE1
$\overline{\text{IPIPE0}}$ / DSO	D[0]	PD[0]	PD[0]	$\overline{\text{CYS}}$ (input)	IPIPE0
		$\overline{\text{IPIPE0}}$]	IPIPE0	NA	IPIPE0
FC[2:0] / $\overline{\text{CS}}$[7:5] (optional)	E[7:5]	PCON[8:7][1]	PCON[8:7][1]]	PCON[8:7][1]	PCON[8:7][1]
		PCON[8:7][1]	PCON[8:7][1]	NA	PCON[8:7][1]
CLKOUT	E[4]	PCON[3][1]	PCON[3][1]	CLKOUT	CLKOUT
		PCON[3][1]	PCON[3][1]	NA	CLKOUT
SIZE	E[3]	SIZE	PE[3]	PE[3]	SIZE
		SIZE	PE[3]	NA	SIZE
AS	E[2]	AS	PE[2]	$\overline{\text{AS}}$ (input)	AS
		AS	PE[2]	NA	AS
DS	E[1]	DS	PE[1]	$\overline{\text{DS}}$ (input)	DS
		DS	PE[1]	NA	DS
R/$\overline{\text{W}}$	E[0]	R/$\overline{\text{W}}$	PE[0]	R/$\overline{\text{W}}$ (input)	R/$\overline{\text{W}}$
		R/$\overline{\text{W}}$	PE[0]	NA	R/$\overline{\text{W}}$
IRQX / $\overline{\text{BERR}}$	F[7]	IRQX	IRQX	IIRQX (output)	IRQX
		IRQX	IRQX	NA	IRQX
IRQ[6:5,3:1] (optional)	F[6:5,3:1]	IRQ[6:5,3:1]	IRQ[6:5,3:1]	IIRQ[6:5,3:1] (output)	IRQ[6:5,3:1]
		IRQ[6:5,3:1]	IRQ[6:5,3:1]	NA	IRQ[6:5,3:1]
IRQ4	F[4]	IRQ4	IRQ4	IIRQ4 (output)	IRQ4
		IRQ4	IRQ4	NA	IRQ4

Table 3-3 Default Pin Function Immediately after RESET (Continued)

Signal	Port	RESET Pin Function in Each Mode (Background Enabled)			
		Master	Single Chip	SLAM	Emulation
CS4 (optional)	F[0]	PCON[6][1]	PCON[6][1]	PCON[6][1]	PCON[6][1]
		PCON[6][1]	PCON[6][1]	NA	PCON[6][1]
D[15:0]	G/H	D[15:0]	PG[7:0], PH[7:0]	D[15:0]	D[15:0]
		D[15:0]	PG[7:0], PH[7:0]	NA	D[15:0]
BCLK	K[7]	PCON[2][1]	PCON[2][1]	PCON[2][1]	PCON[2][1]
		PCON[2][1]	PCON[2][1]	NA	PCON[2][1]
BWE (optional)	K[6]	PCON[2][1]	PCON[2][1]	PCON[2][1]	PCON[2][1]
		PCON[2][1]	PCON[2][1]	NA	PCON[2][1]
BOE (optional)	K[5]	PCON[2][1]	PCON[2][1]	PCON[2][1]	PCON[2][1]
		PCON[2][1]	PCON[2][1]	NA	PCON[2][1]
BAA (optional)	K[4]	PCON[2][1]	PCON[2][1]	PCON[2][1]	PCON[2][1]
		PCON[2][1]	PCON[2][1]	NA	PCON[2][1]
LBA	K[3]	PCON[2][1]	PCON[2][1]	PCON[2][1]	PCON[2][1]
		PCON[2][1]	PCON[2][1]	NA	PCON[2][1]
BREQ	K[2]	PCON[1][1]	PCON[1][1]	$\overline{\text{BREQ}}$ (input)	BREQ
		PCON[1][1]	PCON[1][1]	NA	BREQ
BTACK	K[1]	PCON[1][1]	PCON[1][1]	$\overline{\text{BTACK}}$ (output)	BTACK
		PCON[1][1]	PCON[1][1]	NA	BTACK
DTACK	K[0]	PCON[14][1]	PCON[14][1]	$\overline{\text{DTACK}}$ (output)	DTACK
		PCON[14][1]	PCON[14][1]	NA	DTACK
$\overline{\text{RSTOUT}}$ /CSE0	CSE0	RSTOUT	RSTOUT	RSTOUT	$\overline{\text{RSTOUT}}$ /CSE0
		RSTOUT	RSTOUT	NA	$\overline{\text{RSTOUT}}$ /CSE0

NOTES:
1. PCON value can be overridden by driving external pin(s) at reset NA: Not Allowed mode combinations.

3.1.5.3 PLL Mode Selection

The PLL mode is determined at reset. The state of V_{DDSYN} is sampled on the rising edge of RESET to determine whether the PLL provides the internal BIM clock or whether the clock is provided from an external clock source. The state of the $\overline{\text{LBA}}$ pin determines whether the on-chip PLL operates in normal PLL mode or whether it operates in 1:1 mode. Refer to **3.4 Clocks** for more details.

3.1.5.4 Background Debug Mode Selection

When $\overline{\text{BKPT}}$ is asserted during reset, background debug mode is enabled. Background mode can be entered if the $\overline{\text{BKPT}}$ pin is asserted during a bus cycle, providing that the background mode is enabled, the BGND instruction is executed, a double bus fault occurs, or an internal peripheral asserts the $\overline{\text{IBKPT}}$ signal.

When background debug mode is entered, certain pin functions are required. These pin functions are noted in the colored areas of Table 3-3.

In background mode, FREEZE is asserted and the BIM response is determined by the FRZ bits in the MCR. Based on these encodings, the software watchdog, the real-time clock, and the prescaler may stop counting. In addition, all BIM registers are accessible by the CPU. If a register contains any write-once bits, these bits are readable and writable during FREEZE. When FREEZE is negated, the BIM resumes operation based on the current values contained in its registers. The mask-programmable options are summarized in Table 3-4.

Table 3-4 Customer-Specified Mask-Programmable Options

Option	Register
BCS — BAA function	PCON[15]
Port K[0]/ $\overline{\text{DTACK}}$ function	PCON[14]
Configuration of port C / A[22:16] function	PCON[13:11]
Boot device size	PCON[10]
A[23] / port C[7] function	PCON[9]
FC[2:0] / CS7 — CS5 / port E[7:5] function	PCON[8:7]
CS4 / port F[0] function	PCON[6]
CS3-CS1 / port D[7:5] function	PCON[5]
Normal PLL mode or 1:1 mode	PCON[4]
Port E[4] / CLKOUT pin function	PCON[3]
Burst chip select / port K[7:3] function	PCON[2]
Port K[2:1]/ $\overline{\text{BREQ}}$, $\overline{\text{BTACK}}$ function	PCON[1]
Master mode / single chip mode	PCON[0]

3.1.5.5 Default Configuration Override During RESET

While the BIM is held in reset, data bus pins D[11:0] are configured as inputs with weak pull-ups or pull-downs to reflect the state of the MCR bits or PCON bits corresponding to each D[i] input. D[15:12] have internal pull-ups as in previous integration modules. Default reset values for the PCON register can be overridden by driving the PCON override pins to predefined logic levels during reset as shown in Table 3-5. Please refer to 3.1.12 Port Configuration Shadow Register (PCON) for complete description of all the default reset options.

Table 3-5 Default Configuration During RESET

Pin(s) Affected	RESET Shadow Bit	Override Pin	Effect of Override Pin High During RESET	Effect of Override Pin Low During RESET
All ports	VDD, VDD, PCON[0]	D[10,1,0]	State of override pins 1 1 1 1 1 0 1 0 1 1 0 0 0 x x	Mode selected Master mode Single chip mode Master test mode (MISR enabled) Slave access mode Emulator mode
BCS	PCON[15]	BOE	$\overline{BAA}$ = BAA waveform	$\overline{BAA}$ = HP$\overline{CE}$
A[22:16] / port C[6:0]	PCON[13:11]	D[11,4,2]	State of Override Pins 0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 1 1 1	Configuration of Pins Port C[6:0] Port C[6:1], A[16] Port C[6:2], A[17:16] Port C[6:3], A[18:16] Port C[6:4], A[19:16] Port C[6:5], A[20:16] Port C[6], A[21:16] A[22:16]
Boot device size	PCON[10]	SIZE	8-bit boot device	16-bit boot device
A[23] / port C[7]	PCON[9]	D[9]	A[23]	Port C[7]
FC[2:0] / CS7 – CS5 / port E[7:5]	PCON[8:7]	D[8:7]	State of override pins 0 0 1 0 1 1	Configuration of pins port E[7:5] FC[2:0] CS7 - CS5
CS4 / port F[0]	PCON[6]	D[6]	CS4	Port F[0]
CS3-CS1/ port D[7:5]	PCON[5]	D[5]	CS3 – CS1	Port D[7:5]
Internal / external clock source	NA	V_{DDSYN}	PLL	External clock
PLL Mode (If V_{DDSYN} = 1)	PCON[4]	LBA	Normal PLL mode	1:1 PLL mode
CLKOUT/Port E[4]	PCON[3]	D[3]	CLKOUT	Port E[4]
Burst chip select/port K[7:3]	PCON[2]	BREQ	BCLK, BWE, BOE, BAA, LBA	Port K[7:3]
$\overline{BTACK}$, $\overline{BREQ}$/ port K[2:1]	PCON[1]	BTACK	$\overline{BTACK}$, $\overline{BREQ}$	Port K[2:1]
$\overline{DTACK}$/ port K[0]	PCON[14]	DTACK	DTACK	Port K[0]
MCR[9]	PCON[0]	D[0]	Master mode	Single chip mode
BKPT	NA	BKPT	Background mode disabled	Background mode enabled

3.1.6 EBR Operation

The external bus may be granted to an alternate bus master in master, emulation, or background debug operation mode. The external master cannot access internal MCU resources.

3.1.7 Interrupt Operation

When there is a pending interrupt at an instruction boundary, and no higher priority exceptions are pending, the CPU initiates interrupt processing for that interrupt priority level (IPL). The CPU performs an internal interrupt acknowledge cycle (IACK) to fetch the vector number corresponding to the interrupt service routine. All modules with pending interrupt requests at the interrupt priority level arbitrate for service. If no module arbitrates for the IACK and the IACK cycle is terminated with BERR, a spurious interrupt exception will be taken.

BIM interrupt sources are:

- Real-time counter/software watchdog counter (RTC/SWC)
- Port F edge-detect interrupt logic, and
- the IRQ7-1 pins

Since the IRQ pins may assert an interrupt at the same priority level as the RTC/SWC and/or the port F edge-detect logic, a priority scheme gives the highest priority to the RTC/SWC, followed by the interrupt pins, and then the port F edge-detect logic. An interrupt from a lower priority source remains pending until the next allowable interrupt time.

The RTC/SWC and the port F edge-detect logic have programmable interrupt vector registers that provide the interrupt vector during an IACK cycle.

The BIM supports both autovector and external vector sources for interrupt requests indicated by the external IRQ pins. The BIM can terminate external IACK cycles by asserting IAVEC, indicating that the autovector corresponding to the interrupt priority level (IPL) should be used. Alternately, the IACK cycle can be terminated by providing an external interrupt vector, and asserting $\overline{\text{DTACK}}$. $\overline{\text{DTACK}}$ can be generated by an external device or by a chip select.

An autovector response is selected for a specific IRQ pin by setting the corresponding AV bit in the IACK response register. When an IACK cycle is executed for the IPL corresponding to the level of the IRQ pin, the BIM asserts IAVEC to cause the CPU to terminate the IACK cycle.

The BIM executes external IACK cycles when the AV bit is negated for the respective IPL. Chip selects can be programmed to assert during IACK cycles and to provide cycle termination. See **3.6.4 Asynchronous Chip Select Operations**.

3.1.8 LPSTOP Operation

To facilitate low power standby operation, the BIM is designed to operate with CPUs which support a low power STOP instruction (LPSTOP). When the CPU executes the LPSTOP instruction, the BIM decodes a CPU space type 3 cycle. A copy of the mask

priority level is written to the BIM, and an external indication of LPSTOP execution is available through chip select option programming. The BIM terminates the LPSTOP cycle by asserting internal DTACK. Once LPSTOP mode is entered, the only way to restart the system is by reset or by any interrupt request that is higher than the interrupt mask priority level. The interrupt source can be the port F I/O pins, IRQ pins (level or edge), or either modulus counter in the system protection sub-module. See **3.4.5 LPSTOP Operation** for information on programming which system clocks are disabled in LPSTOP and other options.

3.1.9 Emulator Support

The BIM contains support for emulator logic which can be used to faithfully replicate on-chip digital I/O ports externally. In emulation mode, accesses to internal port registers become external accesses to the port replacement logic. Access to port replacement logic is transparent to the application software, including access time. All other BIM register accesses are unaffected in emulation mode.

Emulation mode is entered by pulling D[10] low during reset. To ensure proper operation of the emulator logic, the LOAD control bit in the MCR is overridden, forcing the pad drivers to full load.

While in emulation mode, special emulator chip selects (CSE1, CSE0) are driven externally to allow internal accesses to be tracked by external hardware. See **Table 3-6**. CSE1 is time-multiplexed with FREEZE and CSE0 is time-multiplexed with the RSTOUT pin. The special emulation chip selects are asserted with the same timing as address strobe (AS). Emulation mode chip selects indicate whether the bus cycle is an external access, external port register access, or internal module control block access.

3.1.9.1 External Port Replacement Indication

In emulation mode, all port A, B, C, D, E, G, H, and K registers are mapped externally **(IAACK is not driven by the BIU, but IDTACK is driven in two clocks to improve external timing)**. The states of MODMAP and IFC[2:0] = Z01 are included in the address decode. CSE1 and CSE0 = 10 whenever any emulated port registers are addressed without privilege violation (0xY FFA10 — 0xYF FA2F). **The BIM does not respond to these accesses, other than to terminate the cycle in two clocks, so that external emulator logic can respond**. The CSE pins indicate external port replacement access on byte accesses to individual registers, as well as word accesses to data and data direction register pairs.

When the BIM is in emulation mode, port accesses to supervisor only registers (FC = 101) are checked by the BIM only to Z01 functionality. Emulation logic should further qualify emulated registers for supervisor only access.

3.1.9.2 Internal Module Control Block Indication

Accesses to the 32-Kbyte address block which contains the module control blocks (0xYF 8000 — 0xYF FFFF), exclusive of those registers listed in **3.1.9.1 External**

Port Replacement Indication, are indicated by CSE1 and CSE0 = 00. The lower 15 bits of the address bus indicates the register accessed within the 32-Kbyte block.

3.1.9.3 Other Access Indication

In previous integration modules, the external DTACK pin could not be driven on a cycle by cycle basis to terminate two clock external cycles. Due to this limitation, ICSM technology was developed to cope with terminating two clock external emulated cycles. The BIM allows DTACK, BTACK, and BERR to terminate two clock external cycles. The BIM does not support ICSM technology.

To emulate internal modules in BIM based systems, the emulated module is simply disabled allowing its cycles to go external to access emulation hardware.

If an access is not covered by the encodings specified in the previous sections, CSE1 and CSE0 will equal 11.

Table 3-6 Emulator Mode Chip Select Summary

CSE1	CSE0	Indication in Emulator Mode
1	1	All access not covered by CSE encodings = {10, 00}
1	0	Access to emulated port register in emulator mode
0	1	Reserved
0	0	Access to 32-Kbyte module control block, not an emulated port register.

3.1.10 Module Mapping

All of the MCU internal module registers occupy a 32-Kbyte block of memory. This 32-Kbyte register block is movable to one of two locations in the map, starting at 0x7F 8000 or 0xFF 8000, by programming the module mapping (MM) bit in the BIM module configuration register. This allows the user the choice of placing the module registers in the upper short addressing range (0xFF 8000 — 0xFF FFFF), or reserving the entire short addressing range for external devices.

The BIM memory map occupies 128 bytes. Figure 3-5 shows the decoding of the address lines for the BIM internal registers as follows: A[23] is compared to the MM line on the IMB, A[22:15] must be high, A[14:7] must match the assigned value for the module decode as specified in the device control and A[6:0] must match the individual register decodes as detailed in Figure 3-5.

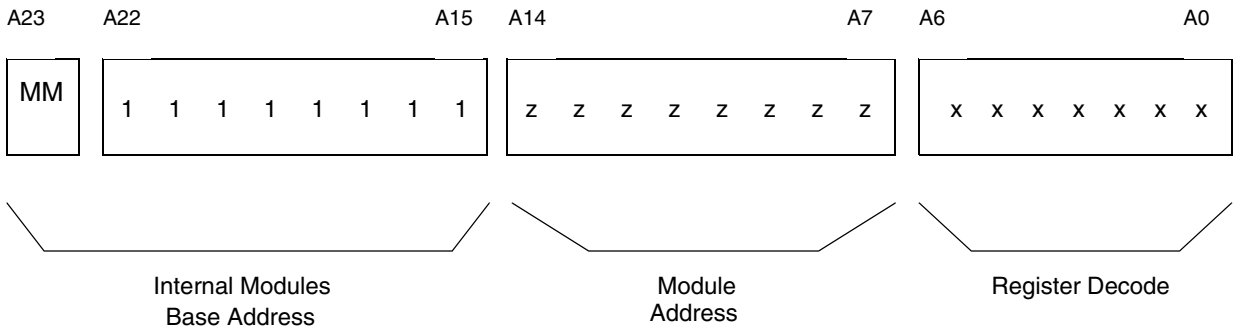


Figure 3-5 BIM Module Address Decoding

NOTE

Even though some CPUs have an internal address bus size other than 24 bits, the BIM is implemented to operate with a 24-bit address bus. The 24-bit (16-Mbyte) module memory map is replicated 256 times when viewed from a CPU with a 32-bit program counter.

3.1.10.1 BIM Memory Map

The BIM memory map occupies 128 bytes of address space. The address space is designated as supervisor-only data space or as supervisor/unrestricted data space. Some CPUs, such as the CPU32 family, support a supervisor state which allows access to all registers and a user state which restricts access to some registers. Function code line two (IFC2) on the IMB is used to determine whether an access is in supervisor or unrestricted mode. Most of the registers in the BIM must be accessed while in supervisor data space. Some registers, however, (refer to Z in the function code column in **Figure 3-5**) may be programmed to allow access in either supervisor or unrestricted mode. In MCU implementations which do not support separate supervisor and unrestricted spaces, IFC2 should be forced to always be asserted.

When the BIM is in emulation mode, normal port accesses to supervisor only registers (FC = 101) are checked only as Z01.

Table 3-7 shows the organization of all registers in the BIM. For more information about individual registers refer to the sub-module shown in the right column. Shaded areas of the memory map are reserved. Writes to reserved locations are ignored and reads return zeros.

Table 3-7 BIM Memory Map

Address	FC	15 8	7 0	Sub Module
0xYF FA00	101	MODULE CONFIGURATION REGISTER (MCR)		BIU
0xYF FA02	101	MODULE TEST REGISTER (MTR)		—
0xYF FA04	101	BIM TEST REGISTER (BIMTR)	MODULE DISABLE REGISTER (MDR)	—
0xYF FA06	101			
0xYF FA08	101	CLOCK SYTHESIZER CONTROL (SYNCR)		CLOCK
0xYF FA0A		CLOCK SYTHESIZER STATUS (SYNST)	RESET STATUS (RSR)	CLOCK
0xYF FA0C				
0xYF FA0E	Z01	PORT/CLOCK CONFIGURATION SHADOW REGISTER (PCON)		
0xYF FA10	Z01	PORT A OUTPUT DATA REGISTER (PORTA)	PORT B OUTPUT DATA REGISTER (PORTB)	PORT
0xYF FA12	Z01	PORT A PIN DATA REGISTER (POR-TAP)	PORT B PIN DATA REGISTER (PORTBP)	—
0xYF FA14	Z01		PORT A/B DATA DIRECTION (DDRAB)	—
0xYF FA16				—
0xYF FA18	Z01	PORT C OUTPUT DATA (PORTC)	PORT D OUTPUT DATA (PORTD)	—
0xYF FA1A	Z01	PORT C PIN DATA REGISTER (PORTCP)	PORT D PIN DATA REGISTER (PORTDP)	—
0xYF FA1C	Z01	PORT C DATA DIRECTION (DDRC)	PORT D DATA DIRECTION (DDRD)	—
0xYF FA1E	101	PORT C PIN ASSIGNMENT (PCPAR)	PORT D PIN ASSIGNMENT (PDPAR)	—
0xYF FA20	Z01	PORT K OUTPUT DATA (PORTK)	PORT E OUTPUT DATA REGISTER (PORTE)	—
0xYF FA22	Z01	PORT K PIN DATA REGISTER (PORTKP)	PORT E PIN DATA REGISTER (PORTEP)	—
0xYF FA24	Z01	PORT K DATA DIRECTION (DDRK)	PORT E DATA DIRECTION (DDRE)	—
0xYF FA26	101	PORT K PIN ASSIGNMENT (PKPAR)	PORT E PIN ASSIGNMENT (PEPAR)	—
0xYF FA28	Z01	PORT G OUTPUT DATA REGISTER (PORTG)	PORT H OUTPUT DATA REGISTER (PORTH)	—
0xYF FA2A	Z01	PORT G PIN DATA REGISTER (PORTGP)	PORT H PIN DATA REGISTER (PORTHP)	—
0xYF FA2C	Z01	PORT G DATA DIRECTION (DDRG)	PORT H DATA DIRECTION (DDRH)	—
0xYF FA2E				—
0xYF FA30	Z01		PORT F OUTPUT DATA (PORTF)	—
0xYF FA32	Z01		PORT F PIN DATA REGISTER (PORTFP)	—
0xYF FA34	Z01		PORT F DATA DIRECTION (DDRF)	—
0xYF FA36	101	PORT F PIN ASSIGNMENT (PFPAR)		—
0xYF FA38	101		PORT F EDGE FLAGS (PORTFE)	—
0xYF FA3A	101	PORT F IACK RESPONSE (PFIACK)	PORT F EDGE-DETECT ENABLE (PFEER)	—
0xYF FA3C	101	PORT F LEVEL (PFLVR)	PORT F INTERRUPT VECTOR (PFIVR)	—
0xYF FA3E				

Table 3-7 BIM Memory Map (Continued)

Address	FC	15 8	7 0	Sub Module
0xYF FA40	101	TEST MODULE MASTER SHIFT A (MSRA)		TEST
0xYF FA42	101	TEST MODULE MASTER SHIFT B (MSRB)		—
0xYF FA44	101	TEST MODULE SHIFT COUNT A (SCRA)	TEST MODULE SHIFT COUNT B (SCRB)	—
0xYF FA46	101	TEST MODULE REPETITION COUNTER (REPS)		—
0xYF FA48	101	TEST MODULE CONTROL REGISTER (CREG)		—
0xYF FA4A	101	TEST MODULE DISTRIBUTED REGISTER (DREG)		—
0xYF FA4C		BURST CHIP SELECT OPTION REGISTER 2 (BCSOR2)		BCS
0xYF FA4E				
0xYF FA50	101	SYSTEM PROTECT CONTROL (SYPCR)		SYSPROT
0xYF FA52	101	TIMER CONTROL (TIC)	TIMER INTERRUPT VECTOR (TIV)	—
0xYF FA54	101		SOFTWARE SERVICE (SWS)	—
0xYF FA56	101	PRESCALER (READ-ONLY) (PRE)		SYSPROT
0xYF FA58	101	SOFTWARE WATCHDOG INTERVAL REGISTER (SWI)		—
0xYF FA5A	101	REAL-TIME PERIOD REGISTER (RTP)		—
0xYF FA5C	101	SOFTWARE WATCHDOG INTERVAL TIMER OPERATION REGISTER (SWIT)		—
0xYF FA5E	101	REAL-TIME DOWNCOUNTER (RTDC) (READ-ONLY)		—
0xYF FA60	101	CHIP SELECT BASE ADDRESS REGISTER 1 (CSBAR1)		CHIP SEL
0xYF FA62	101	CHIP SELECT OPTION REGISTER 1 (CSOR1)		—
0xYF FA64	101	CHIP SELECT BASE ADDRESS REGISTER 2 (CSBAR2)		—
0xYF FA66	101	CHIP SELECT OPTION REGISTER 2 (CSOR2)		—
0xYF FA68	101	CHIP SELECT BASE ADDRESS REGISTER 3 (CSBAR3)		—
0xYF FA6A	101	CHIP SELECT OPTION REGISTER 3 (CSOR3)		—
0xYF FA6C	101	CHIP SELECT BASE ADDRESS REGISTER 4 (CSBAR4)		—
0xYF FA6E	101	CHIP SELECT OPTION REGISTER 4 (CSOR4)		—
0xYF FA70	101	CHIP SELECT BASE ADDRESS REGISTER 5 (CSBAR5)		—
0xYF FA72	101	CHIP SELECT OPTION REGISTER 5 (CSOR5)		—
0xYF FA74	101	CHIP SELECT BASE ADDRESS REGISTER 6 (CSBAR6)		—
0xYF FA76	101	CHIP SELECT OPTION REGISTER 6 (CSOR6)		—
0xYF FA78	101	CHIP SELECT BASE ADDRESS REGISTER 7 (CSBAR7)		—
0xYF FA7A	101	CHIP SELECT OPTION REGISTER 7 (CSOR7)		—
0xYF FA7C	101	BURST CHIP SELECT BASE (BCSBAR)		—
0xYF FA7E	101	BURST CHIP SELECT OPTION REGISTER (BCSOR)		—

3.1.10.2 Module Configuration Register (MCR)

This register controls the BIM configuration. The MCR is reset by SRESET.

MCR — Module Configuration Register **0xYF FA00**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
LOAD	FRZ		SUPVA	MODE			SHEN	SPEED	MM	RESERVED		IARB			
RESET:															
—[1]	1	1	1	1	1	1	—[2]	0	1	0	0	1	1	1	1

NOTES:
1. 0 = Master and singlechip modes, 1 = All other modes
2. 0 = All other modes, 1 = If in emulation mode

Table 3-8 MCR Bit Descriptions

Bit(s)	Name	Description
15	LOAD	Pad driver LOAD. This bit controls the drive strength of all pad output drivers by selecting whether they will be at full or partial drive strength. If IFREEZE is asserted, the bit is write always in all modes. See **Table 3-9** for LOAD bit write accessibility. 0 = Drivers will be at partial drive strength, resulting in a reduction in power consumption 1 = Drivers will be at full strength, allowing the pad to drive the maximum capacitive load
14:13	FRZ	Freeze bit. These bits control how the system protection submodule responds to the assertion of the IFREEZE line on the IMB. See for the FREEZE fields. 00 = Counters ignore the assertion of the IFREEZE line 01 = SWDOG, RTC, prescaler disabled 10 = Bus monitor disabled 11 = Bus monitor, SWDOG, RTC, prescaler disabled
12	SUPVA	Supervisor data space. Used to enable/disable access protection to some of the BIM registers. Most of the registers in the BIM can only be accessed while in supervisor mode. Some register accesses are controlled by the SUPVA bit to allow either supervisor-only access or unrestricted access. Privilege violations to supervisor restricted registers will result in the generation of AACK and DTACK. A read access will return a data value equal zero, write data will be ignored. 0 = Access to registers is unrestricted 1 = Registers are restricted to supervisor-only access
11:9	MODE	Mode configuration. These bits determine the BIM operating mode. 0XX = Double-buffered mode 100 = Slave access mode 110 = Single chip mode 111 = Master mode
8	SHEN	Show cycle enable. Used by the EBI to determine whether to drive the external bus during internal transfer (show cycle) operations. Show cycles only affect the data bus and the control signals $\overline{DS}$. Even if show cycles are disabled, the address information is always driven by the BIM unless mastership of the external bus has been relinquished ($\overline{EBR}$ asserted). In emulator mode, the SHEN bit is forced to a 1. 0 = Show cycles disabled 1 = Show cycles enabled
7	SPEED	I/O pad speed. Controls the rise time of MCU pins configured as a digital outputs. When SPEED is cleared digital output pins, that support a slow rise time to minimize EMI, transistion slowly. When SPEED is set digital outputs are configured to transition in their “fast mode”. The actual rise times (slow/fast) is dependent on the pad type.
6	MM	Module map. etermines the position of the MMF internal modules within the overall device memory map. This bit is driven onto the internal bus and A[23] must match this value for an internal module to be selected by a bus cycle. 0 = Internal modules are located at 0x7FF000 – 0x7FFFFF 1 = Internal modules are located at 0xFFF000 – 0xFFFFFF

Table 3-8 MCR Bit Descriptions (Continued)

Bit(s)	Name	Description
5:4	—	Reserved
3:0	IARB	Interrupt arbitration. Used to assign an interrupt arbitration priority level for the BIM. The BIM arbitrates on the IMB for interrupt servicing based on its IARB value. When interrupts are enabled, the IARB field must be set to a non-zero value. The lowest priority IARB number is 0001 and the highest priority IARB number is 1111. The interrupt sources in the BIM are: port F interrupt logic and the SWDOG or RTC counters in the system protection submodule.

Table 3-9 LOAD Bit Write Accessibility:

Master	Single Chip	Emulation	MFTM, ETM = 0	MFTM, ETM = 1	SLAM, ETM = 0	SLAM, ETM = 1
Write once	Write once	Read only	Read only	Read/Write	Write once	Read/Write

3.1.11 Module Disable Register (MDR)

MDR — Module Disable Register **0xYF FA04**

MSB 7	6	5	4	3	2	1	LSB 0
STP7 (STPCPU)	STP6	STP5	STP4	STP3	STOPCS	STOPTEST	STOPSYS-PROT
RESET:							
0	0	0	0	0	0	0	0

The module disable register (MDR) is used to disable the clocks to IMB modules during LPSTOP. The MDR also contains bits to disable the clocks in the chip select, test sub-module, and system protection sub-modules. The MDR is reset by SRESET.

Table 3-10 MDR Bit Descriptions

Bit(s)	Name	Description
7:5	STP[7:5]	Stop IMB module clock. By convention STP7 is assigned to control the CPU clock and is historically aliased as STPCPU. The STP6 and STP5 bits are used for derivative-specific clock control. During LPSTOP, each STPx bit controls the assertion of a corresponding signal on the IMB. $\overline{\text{CLKDIS}}$ [4:2] are controlled by STP7 (STPCPU), STP6 and STP5, respectively. If a module is connected to an asserted CLKDIS signal, the module will disable its CLOCK signal to achieve the lowest power consumption. To support the use of the STP bits in the MDR to selectively disable the SYSCLKs to each IMB module, the STICLK bit in the SYNCR is set to zero to allow the EXTCLK, BIMCLKs and SYSCLKs to continue to function during LPSTOP. MDR[7:5] are cleared by SRESET or by exiting LPSTOP. 0 = If LPSTOP is executed, the corresponding $\overline{\text{CLKDIS}}$[4:2] signal remains negated 1 = If LPSTOP is executed, the BIM asserts the $\overline{\text{CLKDIS}}$[4:2] signal to disable the module
4:3	STP[4:3]	Stop IMB module clock. During MCU operation, STP4 and STP3 controls the assertion of a corresponding signal on the IMB to statically control an IMB module's clocks. $\overline{\text{CLKDIS}}$[1:0] are controlled by STP4, and STP3, respectively. If a module is connected to an asserted CLKDIS signal, the module will gate off its CLOCK signal to achieve the lowest power consumption. 0 = The corresponding $\overline{\text{CLKDIS}}$[1:0] signal remains negated 1 = The BIM asserts the $\overline{\text{CLKDIS}}$[1:0] signal to disable the module

Table 3-10 MDR Bit Descriptions (Continued)

Bit(s)	Name	Description
2	STOPCS	Stop chip selects. Used to enable/disable access protection to some of the BIM registers. Most of the registers in the BIM can only be accessed while in supervisor mode. Some register accesses are controlled by the SUPVA bit to allow either supervisor-only access or unrestricted access. Privilege violations to supervisor restricted registers will result in the generation of AACK and DTACK. A read access will return a data value equal zero, write data will be ignored. 0 = Chip select logic is enabled 1 = Chip select logic is not enabled
1	STOP-TEST	Stop test. This control bit directs BIM logic to disable clocks to the system protection submodule. While the STOPSYSPROT bit is set, the system protect submodule does not function. 0 = The test submodule logic is enabled 1 = The test submodule logic is not enabled
0	STOP-SYS-PROT	Stop system protection. This control bit directs BIM logic to disable clocks to the system protection submodule. 0 = Not possible to read or write the system protect register 1 = Possible to read or write the system protect register

3.1.12 Port Configuration Shadow Register (PCON)

PCON — BIM Port Configuration Shadow Register **0xYF FA0E**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
BAA	DTACK	A[22:16]			BOOT SIZE	A[23]	$\overline{FC}$[2:0]		CS4	CS3-CS1	PLL REF	CLK-OUT	BURST CS	BTACK BREQ	MODE
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The port configuration shadow register is a read-only, mask programmable register that determines the default pin functions for certain operating modes and the default clock configuration at reset. The reset values of the corresponding bits and fields in the Port C, D, E, F, and K pin assignment registers are affected, see **Table 3-11**. Not all bits in the port pin assignment registers have corresponding shadow bits in PCON.

NOTE

PCON programming must be consistent with pin availability. If a pin does not exist on the device, PCON bits should be programmed to configure the pin function as digital I/O out of reset, since EBI operation may be affected by the presence or absence of these pins.

For single chip applications the PCON value must specify digital I/O options for all port pins. PCON[15] is a don't care if PCON[2] = 0 (Port K[7:3] is digital I/O].

Table 3-11 Port Configuration Shadow Register (PCON)

Bit(s)	Register	Pin(s) Affected	State	Pin Function
PCON[15]	BCSOR2	BAA	1 0	BAA HPCE
PCON[14]	PKPAR[0]	DTACK	0 1	PK[0] DTACK
PCON[13:11]	PCPAR[7:4]	A[22:16]/ PC[6:0]	0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 1 1 1	Port C[6:0] Port C[6:1], A[16] Port C[6:2], A[17:16] Port C[6:3], A[18:16] Port C[6:4], A[19:16] Port C[6:5], A[20:16] Port C[6], A[21:16] A[22:16]
PCON[10]	BCSOR1	—	0 1	16-bit boot device 8-bit boot device
PCON[9]	PCPAR[7]	A[23] / port C[7]	1 0	A[23] Port C[7]
PCON[8:7]	PEPAR[7:6]	FC[2:0] / $\overline{CS7}$ – $\overline{CS5}$ / port E[7:5]	1 0 1 1 0 0	FC[2:0] $\overline{CS7}$ - $\overline{CS5}$ Port E[7:5]
PCON[6]	PFPAR[1:0]	$\overline{CS4}$ / port F[0]	1 0	CS4 Port F[0]
PCON[5]	PDPAR[7:5]	$\overline{CS3}$-$\overline{CS1}$ / port D[7:5]	1 0	CS3-CS1 Port D[7:5]
PCON[4]	SYNST[15]	—	1 0	Normal PLL mode 1:1 PLL mode
PCON[3]	PEPAR[4]	CLKOUT / port E[4]	1 0	CLKOUT Port E[4]
PCON[2]	PKPAR[3]	Burst chip select / port K[7:3]	1 0	Burst chip select Port K[7:3]
PCON[1]	PKPAR[2:1]	$\overline{BTACK}$, $\overline{BREQ}$/ port K[2:1]]	1 0	$\overline{BTACK}$, $\overline{BREQ}$ Port K[2:1]
PCON[0]	MCR[9]	None	1 0	Master mode Single chip mode

3.2 BIM Ports

Many of the pins associated with the BIM may be used for several different functions. Their primary function is to provide an external bus interface for production testing, emulation, and for applications which require off-chip resources to be accessed. When not used for their primary functions, the pins may be used as digital I/O pins. In some cases the pin function is set by the operation mode and the alternate pin functions are not supported.

To facilitate the I/O function, the BIM pins are grouped into 8-bit ports. Each port has associated registers which configure the pins for the desired function, monitor the pins, and control the pins within the port.

When a pin is configured for its primary function, the BIM will enable the 3-V output driver in the pads. When a pin is configured for digital I/O, the BIM will enable the 5-V output driver in the pads.

3.2.1 Port Operation

Pins are configured as digital I/O during reset configuration or by writing to the port's pin assignment register. Ports A, B, G and H do not have pin assignment registers and are configured as digital I/O only if the BIM is placed in single chip mode during reset. Ports C, D, E, F and K have pin assignment registers which allow the user to select between digital I/O or another pin function after reset. The pin assignments at reset are shown in **Table 3-20**.

All digital I/O pins are programmable as either input or output pins via an associated data direction register. In most cases the pins are individually defined as input or output. However, ports A and B are selected as input or output on a whole-port basis.

All ports have both an output data register and a pin data register to monitor and/or control the state of its pins. Writes to the output data register cause IMB data to be stored, and then driven to the corresponding pads of pins programmed as outputs. A read of the output data register returns the current state of this data register, regardless of the actual state of the pins. A read access of a pin data register returns the current state of its corresponding pins, regardless of whether they are input or output. Writes to the pin data registers have no effect.

In addition to the output data and pin data registers, port F has additional registers to support edge or level sensitivity, and interrupt capability. The port F edge-detect flag register indicates that a transition has occurred on an input or output pin. When the corresponding bit in the port F edge-detect I/O interrupt enable register is set, an interrupt is generated at the level specified in the port F interrupt level register. The interrupt vector is defined in the port F edge-detect vector register.

3.2.2 Port Register Access

Internal accesses to all Port A, B, C, D, E, F, G, H and K registers are always two clock accesses. In emulation mode, accesses to these port registers are ignored (IAACK not asserted) allowing the port access to go external so that emulation hardware can satisfy the port access request.

All port registers are byte-addressable and are grouped to allow coherent word access to port data register pairs. Writes to reserved bits in the port registers have no effect and reads return zeros.

3.2.3 Optional Pins

Some port F pins may not be implemented on a particular chip. When the missing pin is programmed to function as an input, the internal input feedback path is tied to V_{SS} and returns a data value of zero if read in the pin data register. Since port F pins are assigned to the IRQ input function after reset in all operation modes except SLAM, the

user must reassign the unimplemented pins to be digital io to avoid spurious interrupts at levels corresponding to the unimplemented pins.

As shown in **Figure 3-6**, special “loopback” logic is implemented to allow better verification of digital I/O and IRQ functionality when optional pins are not present. When a pin is not present but is programmed as an output (primary function outputs and digital outputs), the internal input feedback path normally driven by the pin is driven to the same logic value as the internal output. This loopback logic is controlled by an internal rom plug which indicates whether an optional pin is present in this product derivative. Loopback logic is implemented for the optional pins of ports C, D, E, F, and K.

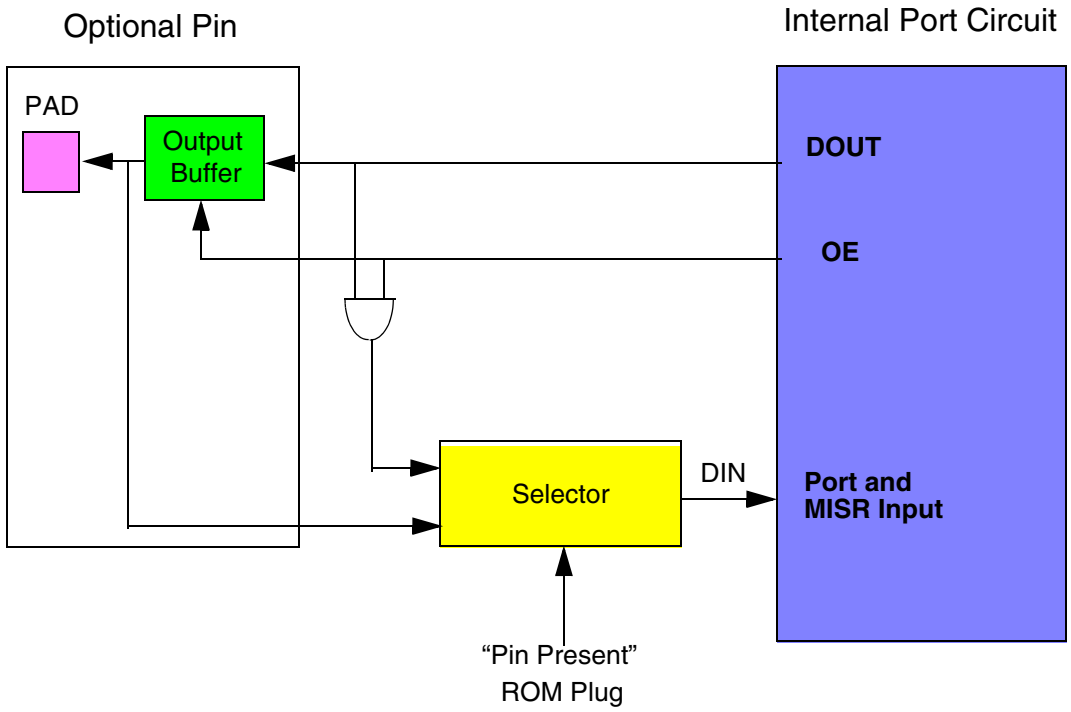


Figure 3-6 Port Loopback Logic on Optional Pins

The MISR is used to improve verification of optional pin’s primary functionality. To improve both observability and testability, all optional outputs are routed to the MISR.

3.2.4 Port A/B Operation

In master mode, MFTM, SLAM, and emulation mode, port A/B functions as external address bus pins A[15:0]. In single chip mode port A/B functions as digital I/O. These pin functions cannot be changed, as there is no pin assignment register. When configured as A[15:0], the BIM will enable the 3-V output drivers in the pads. When configured as digital I/O, the BIM will enable the 5-V output drivers in the pads.

The Port A/B registers are accessible via the IMB unless the BIM is in emulation mode.

Table 3-12 Port A/B Supported Pin Functions

PIN	Master	Single Chip	MFTM	SLAM	Emulation[1]
A[15:0] / PA[7:0], PB[7:0]	A[15:0]	PA[7:0], PB[7:0]	A[15:0]	A[15:0] (input)	A[15:0]

NOTES:
1. Digital I/O pin function provided by port replacement unit

3.2.4.1 Port A/B Data Direction Register (DDRAB)

DDRAB — Port A/B Data Direction Register **0xYF FA14**

MSB 7	6	5	4	3	2	1	LSB 0
RESERVED						DDA	DDB
RESET:							
0	0	0	0	0	0	0	0

Table 3-13 DDRAB Bit Descriptions

Bit(s)	Name	Description
7:2	—	Reserved
1	DDA	Port A data direction. When port A and B are assigned to digital I/O, the pins are configured as either input or output on a whole port basis. 0 = All port A pins are configured as inputs 1 = All port A pins are configured as outputs
0	DDB	Port B data direction. When port A and B are assigned to digital I/O, the pins are configured as either input or output on a whole port basis. 0 = All port B pins are configured as inputs 1 = All port B pins are configured as outputs

3.2.4.2 Port A Output Data Register (PORTA)

PORTA — Port A Output Data Register **0xYF FA10**

MSB 7	6	5	4	3	2	1	LSB 0
PA[15]	PA[14]	PA[13]	PA[12]	PA[11]	PA[10]	PA[9]	PA[8]
RESET:							
0	0	0	0	0	0	0	0

Table 3-14 PORTA Bit Descriptions

Bit(s)	Name	Description
7:0	PA[15:8]	Port A output data. PORTA is used to store the data to be driven on the port A output pins, when the port is configured as output. When read, the current value of the PORTA register, not the port A pins, is returned.

3.2.4.3 Port A Pin Data Register (PORTAP)

PORTAP — Port A Pin Data Register **0xYF FA12**

MSB 7	6	5	4	3	2	1	LSB 0
PA[15]/ PA[7]	PA[14]/ PA[6]	PA[13]/ PA[5]	PA[12]/ PA[4]	PA[11]/ PA[3]	PA[10]/ PA[2]	PA[9]/ PA[1]	PA[8]/ PA[0]
RESET: Not affected by resets.							
0	0	0	0	0	0	0	0

Table 3-15 PORTAP Bit Descriptions

Bit(s)	Name	Description
7:0	PA[7:0]	Port A pin data. When read, PORTAP reflects the current state of the port A pins. Writes to PORTAP have no effect.

3.2.4.4 Port B Output Data Register (PORTB)

PORTB — Port B Output Data Register **0xYF FA10**

MSB 7	6	5	4	3	2	1	LSB 0
PB[7]	PB[6]	PB[5]	PB[4]	PB[3]	PB[2]	PB[1]	PB[0]
RESET:							
0	0	0	0	0	0	0	0

Table 3-16 PORTB Bit Descriptions

Bit(s)	Name	Description
7:0	PB[7:0]	Port B output data. PORTB is used to store the data to be driven on the port B output pins, if the port is configured as an output. When read, the current value of the PORTB register, not the port B pins, is returned.

3.2.4.5 Port B Pin Data Register (PORTBP)

PORTBP — Port B Pin Data Register **0xYF FA12**

MSB 7	6	5	4	3	2	1	LSB 0
A[7]/ PB[7]	A[6]/ PB[6]]	A[5]/ PB[5]	A[4]/ PB[4]	A[3]/ PB[3]	A[2]/ PB[2]	A[1]/ PB[1]	A[0]/ PB[0]
RESET: Not affected by resets.							
0	0	0	0	0	0	0	0

Table 3-17 PORTBP Bit Descriptions

Bit(s)	Name	Description
7:0	A/PB[7:0]	Port B pin data. When read, PORTBP reflects the current state of the port B pins. Writes to PORTBP have no effect.

3.2.5 Port C Operation

In master mode and MFTM, port C can be configured as external address bus pins A[23:16] or as digital I/O. In single chip mode, all port C pins can be configured as external address bus pins A[23:16] or as digital I/O, but only digital I/O is supported.

Table 3-18 indicates the port C pin functions supported for each operation mode. Even when a pin function is not listed as supported for a particular mode, that pin function can still be selected with the port C pin assignment register. However, the pin activity is undefined for that case.

When a port C pin is configured as A[23:16], the BIM will enable the 3-V output driver in the associated pad. When a port C pin is configured as digital I/O, the BIM will enable the 5-V output driver in the associated pad.

Table 3-18 Port C Supported Pin Functions

PIN	Master	Single Chip	MFTM	SLAM	Emulation[1]
A[23] / PC[7]	A[23] or PC[7]	PC[7]	A[23] or PC[7]	PC[7]	A[23]
A[22:16] / PC[6:0]	A[22:16] or PC[6:0]	PC[6:0]	A[22:16] or PC[6:0]	PC[6:0]	A[22:16]

NOTES:
1. Digital I/O pin function provided by port replacement unit.

3.2.5.1 Port C Pin Assignment Register (PCPAR)

PCPAR — Port C Assignment Register **0xYF FA1E**

MSB 7	6	5	4	3	2	1	LSB 0
A[23]/ PC[7]	RESERVED				A[22:16]/ PC[6:0]		
RESET: See Table 3-20.							
0	0	0	0	0	0	0	0

Table 3-19 PCPAR Bit Descriptions

Bit(s)	Name	Description
7	A[23]/PC[7]	Port C configuration. This bit configures the port C[7] pin as either external address bus pin A[23], or digital I/O. When this bit is cleared, port C[7] is configured as digital I/O; when set port C[7] is configured as A[23].
6:3	—	Reserved
2:0	A[22:16]/PC[6:0]	Port C configuration encoding. The bits in this field configure port C[6:0] pins as either external address lines A[22:16] or digital I/O. The value stored in PCPAR[2:0] selects the number of contiguous pins configured as address lines. The address lines selected are contiguous from the least significant pin, A[16], forming a contiguous address space with A[15:0]. The I/O pins are contiguous from the most significant pin, port C[6]. See Table 3-20 for port C pin assignment encodings.

Table 3-20 Port C[2:0] Pin Assignment Encoding

PCPAR[2:0]	Port C I/O Pins	Address Pins
0 0 0	PC[6:0]	None
0 0 1	PC[6:1]	A[16]
0 1 0	PC[6:2]	A[17:16]
0 1 1	PC[6:3]	A[18:16]
1 0 0	PC[6:4]	A[19:16]
1 0 1	PC[6:5]	A[20:16]
1 1 0	PC[6]	A[21:16]
1 1 1	None	A[22:16]

3.2.5.2 Port C Data Direction Register (DDRC)

DDRC — Port Data Direction Register **0xYF FA1C**

MSB 7	6	5	4	3	2	1	LSB 0
PC[7]	PC[6]	PC[5]	PC[4]	PC[3]	PC[2]	PC[1]	PC[0]
RESET:							
0	0	0	0	0	0	0	0

Table 3-21 DDRC Bit Descriptions

Bit(s)	Name	Description
7:0	PC[7:0]	Port C data direction. Port C digital I/O pins can be individually configured as either input or output. The bits in this register control the direction of the port C pin drivers when the pins are configured as I/O. When any bit in this register is set to one, the corresponding pin is configured as an output. When any bit in this register is cleared to zero, the corresponding pin is configured as an input. When the port is not defined to be digital I/O, its corresponding data direction bit has no effect.

3.2.5.3 Port C Output Data Register (PORTC)

PORTC — Port C Output Data Register **0xYF FA18**

MSB 7	6	5	4	3	2	1	LSB 0
PC[7]	PC[6]	PC[5]	PC[4]	PC[3]	PC[2]	PC[1]	PC[0]
RESET:							
0	0	0	0	0	0	0	0

Table 3-22 PORTC Bit Descriptions

Bit(s)	Name	Description
7:0	PC[7:0]	Port C output data. PORTC is used to store the data to be driven on the port C output pins, if the port is configured as output. When read, the current value of the PORTC register, not the port C pins, is returned.

3.2.5.4 Port C Pin Data Register (PORTCP)

PORTCP — Port C Pin Data Register **0xYF FA1A**

MSB 7	6	5	4	3	2	1	LSB 0
A[23]/ PC[7]	A[22]/ PC[6]]	A[21]/ PC[5]	A[20]/ PC[4]	A[19]/ PC[3]	A[18]/ PC[2]	A[17]/ PC[1]	A[16]/ PC[0]
RESET: Not affected by resets.							
0	0	0	0	0	0	0	0

Table 3-23 PORTCP Bit Descriptions

Bit(s)	Name	Description
7:0	A[23:16]/ PC[7:0]	Port C pin data. When read, PORTCP reflects the current state of the port C pins. Writes to PORTCP have no effect.

3.2.6 Port D Operation

When not in emulation mode, port D pins can be selected individually to be either digital I/O, or their primary function as chip select signals and background debug signals. If the BIM is in emulation mode, any access to port D registers is ignored (IAACK not asserted). In emulation mode, port D pins are configured as their primary function.

Table 3-24 indicates the port D pin functions supported for each operation mode. Even when a pin function is not listed as supported for a particular mode, that pin function can still be selected with the port D pin assignment register. However, the pin activity is undefined for that case.

When a port D pin is configured as its primary function, the BIM will enable the 3-V output driver in the associated pad. When a port D pin is configured as digital I/O, the BIM will enable the 5-V output driver in the associated pad.

Table 3-24 Port D Supported Pin Functions

PIN	Master	Single Chip	MFTM	SLAM	Emulation[1]
	Background	Background	Background	Background	Background
$\overline{CS3}$-$\overline{CS1}$ / PD[7:5]	$\overline{CS3}$-$\overline{CS1}$ or PD[7:5]	PD[7:5]	$\overline{CS3}$-$\overline{CS1}$ or PD[7:5]	$\overline{CS3}$-$\overline{CS1}$	$\overline{CS3}$-$\overline{CS1}$
	$\overline{CS3}$-$\overline{CS1}$ PD[7:5]	PD[7:5]	$\overline{CS3}$-$\overline{CS1}$ PD[7:5]	NA	$\overline{CS3}$-$\overline{CS1}$
FREEZE / QUOT / CSE1 / PD[4]	FREEZE or PD[4]	FREEZE or PD[4]	FREEZE or QUOT[2] or PD[4]	FREEZE (input) or QUOT[2]	FREEZE / CSE1[3]
	FREEZE	FREEZE	FREEZE	NA	FREEZE / CSE1[3]
$\overline{BKPT}$ / DSCLK / PD[3]	$\overline{BKPT}$ / DSCLK[3] or PD[3]	$\overline{BKPT}$ / DSCLK[3] or PD[3]	$\overline{BKPT}$ / DSCLK[3] or PD[3]	PD[3]	$\overline{BKPT}$ / DSCLK[3]
	$\overline{BKPT}$ / DSCLK[3]	$\overline{BKPT}$ / DSCLK[3]	$\overline{BPT}$ / DSCLK[3]	NA	$\overline{BKPT}$ / DSCLK[3]
IPIPE2 / PD[2]	IPIPE2 or PD[2]	IPIPE2 or PD[2]	IPIPE2 or PD[2]	PD[2]	IPIPE2
	IPIPE2 or PD[2]	IPIPE2 or PD[2]	IPIPE2 or PD[2]	NA	IPIPE2 or PD[2]
$\overline{IPIPE1}$ / DSI / PD[1]	IPIPE1 or PD[1]	IPIPE1 or PD[1]	IPIPE1 or PD[1]	LATCH (input)	IPIPE1
	DSI	DSI	DSI	NA	DSI
$\overline{IPIPE0}$ / DSO / PD[0]	$\overline{IPIPE0}$ or $\overline{PD[0]}$	$\overline{IPIPE0}$ or $\overline{PD[0]}$	IPIPE0 or $\overline{PD[0]}$	$\overline{CYS}$ (input)	IPIPE0
	DSO	DSO	DSO	NA	DSO

NOTES:
1. Digital I/O pin function provided by port replacement unit.
2. Selected by bit in module test control register (CREG).
3. $\overline{BKPT}$ becomes DSCLK after FREEZE is asserted.

3.2.6.1 Port D Pin Assignment Register (PDPAR)

PDPAR — Port D Pin Assignment Register **0xYF FA1E**

MSB 7	6	5	4	3	2	1	LSB 0
$\overline{CS3}$-$\overline{CS1}$/PD[7:5]			FREEZE/ QOUT CSE1/PD[4]	$\overline{BKPT}$/ $\overline{DSCLK}$/PD[3]	$\overline{IPIPE2}$/ PD[2]	$\overline{PIPE1}$/ DSI/ LATCH/PD[1]	$\overline{PIPE0}$/ DSO/ $\overline{CYS}$/ PD[0]
RESET: See Table 3-3.							
0	0	0	0	0	0	0	0

Table 3-25 PDPAR Bit Descriptions

Bit(s)	Name	Description
7:5	—	Port D pin function. The bits in this register control the function of each pin of port D. 0 = Corresponding pin is configured as a digital I/O pin 1 = Corresponding pin is used for its primary function

3.2.6.2 Port D Data Direction Register (DDRD)

DDRD — Port D Data Direction Register **0xYF FA1C**

MSB 7	6	5	4	3	2	1	LSB 0
PD[7]	PD[6]	PD[5]	PD[4]	PD[3]	PD[2]	PD[1]	PD[0]
RESET:							
0	0	0	0	0	0	0	0

Table 3-26 DDRD Bit Descriptions

Bit(s)	Name	Description
7:0	PD[7:0]	Port D data direction. The bits in this register control the direction of the port D pin drivers when the pins are configured as I/O. When the port is not defined to be digital I/O, its corresponding data direction bit has no effect. 0 = Corresponding pin is configured as an input 1 = Corresponding pin is configured as an output

3.2.6.3 Port D Output Data Register (PORTD)

PORTD — Port D Output Data Register **0xYF FA1A**

MSB 7	6	5	4	3	2	1	LSB 0
PD[7]	PD[6]	PD[5]	PD[4]	PD[3]	PD[2]	PD[1]	PD[0]
RESET:							
1	1	1	1	1	1	1	1

Table 3-27 PORTD Bit Descriptions

Bit(s)	Name	Description
7:0	PD[7:0]	Port D output data. PORTD is used to latch the data to be driven on the port D output pins, if the port is configured as output. When read, the current value of the PORTD register, not the port D pins, is returned.

3.2.6.4 Port D Pin Data Register (PORTDP)

PORTDP — Port D Pin Data Register **0xYF FA1A**

MSB 7	6	5	4	3	2	1	LSB 0
$\overline{CS3}$/PD[7]	$\overline{CS2}$/PD[6]	$\overline{CS1}$/PD[5:]	FREEZE/ QOUT CSE1/PD[4]	$\overline{BKPT}$/ $\overline{DSCLK}$PD[3]	$\overline{IPIPE2}$/ PD[2]	$\overline{PIPE1}$/ DSI/ LATCH/PD[1]	$\overline{PIPE0}$/ $\overline{DSO}$/ $\overline{CYS}$/
RESET: Not affected by RESET.							
0	0	0	0	0	0	0	0

Table 3-28 PORTDP Bit Descriptions

Bit(s)	Name	Description
7:0	—	Port D pin data. When read, PORTDP reflects the current state of the port D pins. Writes to PORTDP have no effect.

3.2.7 Port E Operation

In master mode and MFTM, Port E pins can be selected individually to be either digital I/O, or external bus control/handshake signals. In single chip mode, port E can function as either digital I/O, or external bus control/handshake signals, but only digital I/O is supported. In SLAM, port E pins [7:5] are configured as either $\overline{CS7}$-$\overline{CS5}$ or PC[7:5], since the function codes are supplied on the address pins. In SLAM, pin [4] is typically configured as a clock signal, although the pin can function as PE[4] once the tester has been synchronized. In SLAM, pins [2:0] are configured as cycle control signals. Since the function codes pins are not used in SLAM, these external pins can be selected as either $\overline{CS7}$-$\overline{CS5}$ or PE[7:5]. In emulation mode, port E pins are configured as their primary chip select, bus control, and clock functions.

Table 3-29 indicates the port E pin functions supported for each operation mode. Even when a pin function is not listed as supported for a particular mode, that pin function can still be selected with the port E pin assignment register. However, the pin activity is undefined for that case.

When a port E pin is configured as its primary function, the BIM will enable the 3-V output driver in the associated pad. When a port E pin is configured as digital I/O, the BIM will enable the 5-V output driver in the associated pad.

Table 3-29 Port E Supported Pin Functions

Pin	Master	Single Chip	MFTM	SLAM	Emulation[1]
FC[2:0] / CS7-CS5 / PE[7:5]	FC[2:0] or CS7-CS5 or PE[7:5]	PE[7:5]	FC[2:0] or CS7-CS5 or PE[7:5]	CS7-CS5 or PE[7:5]	CS7-CS5
CLKOUT / PE[4]	CLKOUT or PE[4]	CLKOUT or PE[4]	CLKOUT or PE[4]	CLKOUT or PE[4]	CLKOUT
SIZE / BLOCK PE[3]	SIZE or PE[3]	PE[3]	SIZE or PE[3]	BLOCK (input)	SIZE
AS / PE[2]	AS or PE[2]	PE[2]	AS or PE[2]	AS (input)	AS
DS / PE[1]	DS or PE[1]	PE[1]	DS or PE[1]	DS (input)	DS
R/W / PE[0]	R/W or PE[0]	PE[0]	R/W or PE[0]	R/W (input)	R/W

NOTES:
1. Digital I/O pin function provided by port replacement unit.

3.2.7.1 Port E Pin Assignment Register (PEPAR)

PEPAR — Port E Pin Assignment Register **0xYF FA26**

MSB 7	6	5	4	3	2	1	LSB 0
FC[2:0]/PD[7:5] CS7-5/PE[7:5]		Reserved	CLK-OUT/PE[4]	SIZE/BLOCK /PE[3]	AS/PE[2]	DS/PE[1]	R/W / PE[0]
RESET: See Table 3-3.							
0	0	0	0	0	0	0	0

Table 3-30 PEPAR Bit Descriptions

Bit(s)	Name	Description
7:6	FC[2:0]/ PD[7-5], CS7-5, PE[7:5]	Port E pin function. The bits in this field configure the port E[7:5] pins as either function codes, chip selects, or digital I/O. 00 = PE[7:5] 01 = Reserved 10 = FC[2:0] 11 = CS7-CS5
5	—	Reserved
4:0	—	Port E pin function. The bits in this field control the function of port E[4:0]. 0 = Corresponding pin is configured as an I/O pin 1 = Corresponding pin is configured as its primary function

3.2.7.2 Port E Data Direction Register (DDRE)

DDRE — Port E Data Direction Register **0xYF FA24**

MSB 7	6	5	4	3	2	1	LSB 0
PE[7]	PE[6]	PE[5]	PE[4]	PE[3]	PE[2]	PE[1]	PE[0]
RESET:							
0	0	0	0	0	0	0	0

Table 3-31 DDRE Bit Descriptions

Bit(s)	Name	Description
7:0	PE[7:0]	Port E data direction. The bits in this register control the direction of the port E pin drivers when the pins are configured as I/O. When the port is not defined to be digital I/O, its corresponding data direction bit has no effect. 0 = Corresponding pin is configured as an input 1 = Corresponding pin is configured as an output

3.2.7.3 Port E Output Data Register (PORTE)

PORTE — Port E Output Data Register **0xYF FA20**

MSB 7	6	5	4	3	2	1	LSB 0
PE[7]	PE[6]	PE[5]	PE[4]	PE[3]	PE[2]	PE[1]	PE[0]
RESET:							
0	0	0	0	0	0	0	0

Table 3-32 PORTE Bit Descriptions

Bit(s)	Name	Description
7:0	PE[7:0]	Port E output direction. PORTE is used to latch the data to be driven on the port E output pins, if the port is configured as output. When read, the current value of the PORTE register, not the port E pins, is returned.

3.2.7.4 Port E Pin Data Register (PORTEP)

PORTEP — Port E Pin Data Register **0xYF FA22**

MSB 7	6	5	4	3	2	1	LSB 0
FC[2]/$\overline{CS7}$/ PE[7]	FC[1]/$\overline{CS6}$/ PE[6]	FC[0]/$\overline{CS5}$/ PE[5:]	CLKOUT/ PE[4]	SIZE/PE[3]	$\overline{AS}$/PE[2]	$\overline{DS}$/PE[1]	R/$\overline{W}$ / PE[0]
RESET: Not affected by RESET.							
0	0	0	0	0	0	0	0

Table 3-33 PORTEP Bit Descriptions

Bit(s)	Name	Description
7:0	—	Port E pin data. When read, PORTEP reflects the current state of the port E pins. Writes to PORTEP have no effect.

3.2.8 Port F Operation

This section Table 3-34 indicates the port F pin functions supported for each operation mode. Even when a pin function is not listed as supported for a particular mode, that pin function can still be selected with the port F pin assignment register. However, the pin activity is undefined for that case.

When a port F pin is configured as a level or edge sensitive interrupt, the BIM will enable the 3-V output driver in the associated pad. When a port F pin is configured as an edge detect digital I/O, the BIM will enable the 5-V output driver in the associated pad.

Unlike the other ports, port F registers do respond when the BIM is in emulation mode. The pin function can be changed in emulation mode by accessing the port F pin assignment register.

Table 3-34 Port F Supported Pin Functions

Pin	Master	Single Chip	MFTM	SLAM	Emulation
$\overline{\text{IRQX}}$ / $\overline{\text{BERR}}$ / PF[7]	IRQX or $\overline{\text{BERR}}$ or PF[7]	IRQX or PF[7]	$\overline{\text{IRQX}}$* or $\overline{\text{BERR}}$ or PF[7]	IIRQX (out) or PF[7]	IRQX or $\overline{\text{BERR}}$ or PF[7]
$\overline{\text{IRQ[6:5,3:1]}}$ / PF[6:5,3:1]	IRQ[6:5,3:1] or PF[6:5,3:1]	IRQ[6:5,3:1] or PF[6:5,3:1]	IRQ[6:5,3:1] or PF[6:5,3:1]	IIRQ[6:5,3:1] (out) or PF[6:5,3:1]	IRQ[6:5,3:1] PF[6:5,3:1]
$\overline{\text{IRQ4}}$ / PF[4]	IRQ4 or PF[4]	IRQ4 or PF[4]	IRQ4 or PF[4]	IIRQ4 (out) or PF[4]	IRQ4 or PF[4]
$\overline{\text{CS4}}$ / PF[0]	CS4 or PF[0]	PF[0]	CS4 or PF[0]	CS4 or PF[0]	CS4 or PF[0]

In all operation modes, port F pins can be selected individually to be either digital I/O or their primary function ($\overline{\text{IRQ}}$ or $\overline{\text{CS4}}$). An additional feature of port F is that software may generate both $\overline{\text{IRQ}}$ service requests or digital edge detect interrupt service requests using the loopback logic.

If the SHIRQ bit is set in the BIMTR, port F must be programmed for $\overline{\text{IRQ}}$ operation to allow the internal IIRQ signals to be output. In SLAM, the $\overline{\text{IRQ}}$ pin function is limited to Show $\overline{\text{IRQ}}$ signals, regardless of the SHOW_IRQ bit setting in the BIMTR; the normal $\overline{\text{IRQ}}$ input pin function is not supported.

Table 3-35 Port F Master, Single Chip, MFTM, and Emulation Pin Operation

Pin(s)	Level-Sensitive Interrupt Pin PAR[i+1, i] = 11	Edge-Sensitive Interrupt Pin PAR[i+1, i] = 00	Digital I/O with Edge-Detect PAR[i+1, i] = 01 or 10
$\overline{IRQX}$ / PF[7] (IRQXL = 7)	If the $\overline{IRQX}$ pin is held low for two clocks, the BIM asserts the IMB IRQ7 line. The pin must remain low until the level 7 IACK cycle is initiated. The pin should be returned high before exception processing is completed. Additional IRQ7 level-sensitive interrupts are not recognized until a falling edge is detected to prevent redundant servicing.	If a high-to-low transition is detected on the $\overline{IRQX}$ pin, the BIM latches the condition and asserts the IMB IRQ7 line. The $\overline{IRQX}$ pin is not required to remain low until the level 7 IACK cycle is initiated. The latched condition is cleared when the BIM wins arbitration for the level 7 IACK cycle.	The IRQXL field has no affect when the PFPAR register selects PF[7]. If the edge specified in the PFPAR register for the PF[7] is detected, the BIM sets PORTFE[7]. If PFEER[7] is set, the BIM also asserts the IMB IRQ line corresponding to the priority level specified in PFEL field. The pin is not required to remain asserted. The latched condition is cleared when the user writes a zero to PORTFE[7].
$\overline{IRQX}$ / PF[7] (IRQXL < 7 and IRQXL > 0)	If the IRQXL field in the PFLVR register is less than seven and not zero, and the $\overline{IRQX}$ pin is held low for two clocks, the BIM asserts the IMB IRQ line corresponding to the priority level specified by IRQXL. The pin must remain low until an IACK cycle for that level is initiated and the BIM wins arbitration. The pin should be returned high before exception processing is completed.	If the IRQXL field in the PFLVR register is less than seven and not zero, and a high-to-low transition is detected on the $\overline{IRQX}$ pin, the BIM latches the condition and asserts the IMB IRQ line corresponding to the priority level specified by IRQXL. The $\overline{IRQX}$ pin is not required to remain low until an IACK cycle for that level is initiated.The latched condition is cleared when the BIM wins arbitration for the IACK cycle.	The IRQXL field has no affect when the PFPAR register selects PF[7]. If the edge specified in the PFPAR register for the PF[7] is detected, the BIM sets PORTFE[7]. If PFEER[7] is set, the BIM also asserts the IMB IRQ line corresponding to the priority level specified in PFEL field. The pin is not required to remain asserted. The latched condition is cleared when the user writes a zero to PORTFE[7].
BERR / PF[7] (IRQXL = 0)	If IRQXL = 0, $\overline{IRQX}$ is configured as the $\overline{BERR}$ input pin.	If IRQXL = 0, $\overline{IRQX}$ is configured as the $\overline{BERR}$ input pin.	The IRQXL field has no affect when the PFPAR register selects digital I/O (PF[7]). If the edge specified in the PFPAR register for PF[7] is detected, the BIM sets PORTFE[7]. If PFEER[7] is set, the BIM also asserts the IMB IRQ line corresponding to the priority level specified in PFEL field. The pin is not required to remain asserted. The latched condition is cleared when the user writes a zero to PORTFE[7].

Table 3-35 Port F Master, Single Chip, MFTM, and Emulation Pin Operation

Pin(s)	Level-Sensitive Interrupt Pin PAR[i+1, i] = 11	Edge-Sensitive Interrupt Pin PAR[i+1, i] = 00	Digital I/O with Edge-Detect PAR[i+1, i] = 01 or 10
IRQ[6:1]	The priority level of interrupt request inputs $\overline{\text{IRQ}}$[6:1] is fixed to correspond to the interrupt pin number (e.g., $\overline{\text{IRQ}}$[6] has a fixed priority level of 6). If held low for two clocks, the BIM asserts the IMB IRQ line corresponding to the Interrupt Input pin number.The pin must remain low until an IACK cycle for that level is initiated and the BIM wins arbitration.The pin should be returned high before exception processing is completed.	If a high-to-low transition is detected, the BIM latches the condition and initiates interrupt exception processing at the priority level corresponding to the interrupt input pin number. The pin is not required to remain low. The latched condition is cleared when the BIM recognizes an IACK cycle for the specified priority level and wins the arbitration.	If the edge specified in the PFPAR register for the specific pin is detected, the BIM will set the corresponding flag bit in the PORTFE register. If the corresponding enable bit in PFEER is set, the BIM also asserts the IMB IRQ line corresponding to the priority level specified in PFEL field. The pin is not required to remain asserted. The latched condition is cleared when the user writes a zero to the corresponding flag bit in PORTFE[6:1].
CS4 / PF[0]	If not programmed as a rising or falling edge-detect digital I/O pin, the CS4 pin operates as chip select. Cannot be programmed to operate as a level-sensitive interrupt request pin.	If not programmed as a rising or falling edge-detect digital I/O pin, the CS4 pin operates as chip select. Cannot be programmed to operate as a level-sensitive interrupt request pin.	If the edge specified in the PFPAR register for the PF[0] is detected, the BIM sets PORTFE[0]. If PFEER[0] is set, the BIM also asserts the IMB IRQ line corresponding to the priority level specified in PFEL field. The pin is not required to remain asserted. The latched condition is cleared when the user writes a zero to PORTFE[0].

All edge detect pin functions require the external pin to be asserted for a minimum of one clock plus the asynchronous set-up time. This allows port F circuitry to synchronize and latch the interrupt request to the BIM's internal clock.

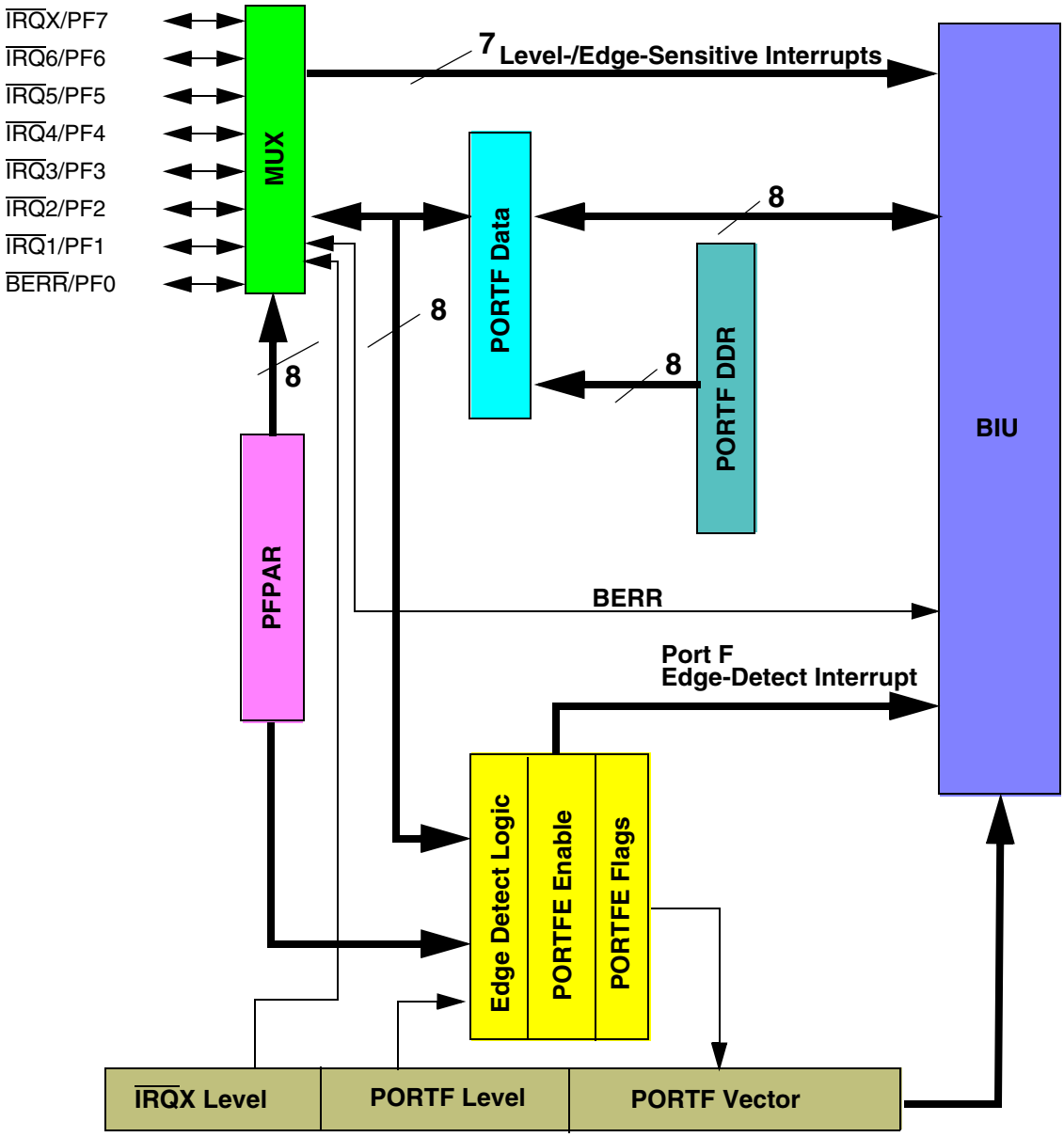


Figure 3-7 PORT F Register Block Diagram

3.2.8.1 Port F Pin Assignment Register (PFPAR)

PFPAR — Port F Pin Assignment Register **0xYF FA36**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
$\overline{\text{IRQX}}$ / $\overline{\text{BERR}}$ / PF[7]		$\overline{\text{IRQ6}}$ / PF[6]		$\overline{\text{IRQ5}}$ /PF[5]		$\overline{\text{IRQ4}}$ /PF[4]		$\overline{\text{IRQ3}}$ /PF[3]		$\overline{\text{IRQ2}}$ /PF[2]		$\overline{\text{IRQ1}}$ /PF[1]		$\overline{\text{CS4}}$ / PF[0]	
RESET: See **Table 3-3**. Note that PCON[6] = 0 selects the 01 (rising edge detect I/O pin) encoding and PCON[6] = 1 selects the 11 ($\overline{\text{CS4}}$) encoding for PFPAR[1:0]. The Level-Sensitive Interrupt Request pin encoding (11) is selected at reset for PFPAR[15:2].															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-36 PFPAR Bit Descriptions

Bit(s)	Name	Description
15:0	—	Port F pin assignment. The eight fields in this register each control the function of a pin in port F. In all operation modes, the field encoding 01 and 10 configure the associated pin as a digital I/O pin with rising or falling edge-detect. In master, single chip, MFTM, and emulation modes, the field encoding 11, configures the corresponding pin as a level-sensitive interrupt request for pins [7:1] and $\overline{\text{CS4}}$ on pin [0]. The field encoding 00, configures the corresponding pin as a falling edge-sensitive interrupt request for pins [7:1] and $\overline{\text{CS4}}$ on pin [0]. To select $\overline{\text{BERR}}$, the field must be set to either 00 or 11 and the IRQXL field in PFLVR must be set to zero. As an alternate pin function for $\overline{\text{IRQX}}$, and $\overline{\text{IRQ}}$[6:1], the internal IMB $\overline{\text{IRQ}}$ lines can be output on these external pins and the value of the internal $\overline{\text{IRQ}}$ lines can also be read in the port F pin data register (PORTFP). In SLAM, the encodings 00 or 11 always select this alternate pin function. In MFTM and in background debug mode, this alternate pin function can be selected by setting the SHIRQ bit in the BIMTR. 00 = Falling edge sensitive interrupt request (pins 7 to 1), $\overline{\text{CS4}}$ (pin 0) 01 = Digital I/O with rising edge detect 10 = Digital I/O with falling edge detect 11 = Level-sensitive interrupt request (pins 7 to 1), $\overline{\text{CS4}}$ (pin 0)

3.2.8.2 Port F Data Direction Register (DDRF)

DDRF — Port F Data Direction Register **0xYF FA34**

MSB 7	6	5	4	3	2	1	LSB 0
PF[7]	PF[6]	PF[5]	PF[4]	PF[3]	PF[2]	PF[1]	PF[0]
RESET:							
0	0	0	0	0	0	0	0

Table 3-37 DDRF Bit Descriptions

Bit(s)	Name	Description
7:0	PF[7:0]	Port F data direction. The bits in this register control the direction of the port F pin drivers when the pins are configured as digital I/O pins. When any bit in this register is set to one, the corresponding pin is configured as an output. When any bit in this register is cleared to zero, the corresponding pin is configured as an input. Unlike the other ports, DDRF also affects port F operation when the PFPAR selects the primary pin function, $\overline{IRQ}$. When $\overline{IRQ}$ is selected and interrupt requests are generated by the $\overline{IRQ}$ pin, the DDRF bit must select input. This is the common programming choice for customer applications. When $\overline{IRQ}$ is selected and interrupt requests are generated by programming the PORTF output data register, the DDRF must select output. When $\overline{IRQ}$ is selected and SHIRQ is programmed in the BIMTR, the DDRF must select output to allow optional Port F pins to be read in PORTFP. Then the loopback logic allows testing of the $\overline{IRQ}$ functionality regardless of whether the pin is present. This is the usual programming for functional test patterns. It is also useful for customer applications that need to generate software interrupts at various interrupt levels. In SLAM, or when the SHIRQ bit is set during MFTM or background debug mode, the DDRF has no affect when PFPAR selects the primary pin function, $\overline{IRQ}$. In this case, the internal IMB $\overline{IRQ}$ lines are reflected on the external pin and can be read in PORTFP. The DDRF[0] has no affect when the CS4 pin function is selected in PFPAR.t

3.2.8.3 Port F Output Data Register (PORTF)

PORTF — Port F Output Data Register **0xYF FA30**

MSB 7	6	5	4	3	2	1	LSB 0
PF[7]	PF[6]	PF[5]	PF[4]	PF[3]	PF[2]	PF[1]	PF[0]
RESET:							
1	1	1	1	1	1	1	1

Table 3-38 PORTF Bit Descriptions

Bit(s)	Name	Description
7:0	PF[7:0]	Port F output data. The PORTF register is used to store the data to be driven on the port F output pins. When read, the current value of the PORTF register, not the PORTF pins, is returned.

3.2.8.4 Port F Pin Data Register (PORTFP)

PORTFP — Port F Pin Data Register **0xYF FA32**

MSB 7	6	5	4	3	2	1	LSB 0
$\overline{IRQ}$X/$\overline{BERR}$/ PF[7]	$\overline{IRQ}$6/PF[6]	$\overline{IRQ}$5/PF[5:]	$\overline{IRQ}$4/PF[4]	$\overline{IRQ}$3/PF[3]	$\overline{IRQ}$2/PF[2]	$\overline{IRQ}$1/PF[1]	CS4/PFE[0]
RESET: Not affected by RESET.							
0	0	0	0	0	0	0	0

Table 3-39 PORTFP Bit Descriptions

Bit(s)	Name	Description
7:0	—	Port F data input. When read, PORTFP reflects the current state of the port F pins. Writes to PORTFP have no effect.

3.2.8.5 Port F Edge-Detect Flag Register (PORTFE)

PORTFE — Port F Edge-Detect Flag Register **0xYF FA38**

MSB 7	6	5	4	3	2	1	LSB 0
EF7	EF6	EF5	EF4	EF3	EF2	EF1	EF0
RESET:							
0	0	0	0	0	0	0	0

Table 3-40 PORTFE Bit Descriptions

Bit(s)	Name	Description
7:0	EF[7:0]	Port F edge detect flag. The port F edge-detect flag register (PORTFE) indicates that the programmed transition has occurred on Port F input or output pins. The edge-detect flag bits in PORTFE are set if the specified edge is detected on the corresponding port F pin, when the corresponding port F pin is configured as an input or output edge-detect pin. If a pin is configured as an $\overline{IRQ}$, $\overline{BERR}$, or $\overline{CS4}$, the flag for that pin is not set on a transition. If a pin transition occurs as a result of changing the PAR encoding from the primary pin function to digital I/O, an edge will be detected. Also, an interrupt can optionally be generated after the proper transition has been detected. An interrupt request is generated whenever the enable bit in the port F edge-detect I/O Interrupt enable register (PFEER) for the corresponding pin is a one, and the port F edge-detect level field in the port F interrupt level register (PFLVR) is non-zero. The bit in the PORTFE, PFEER, and the PFLVR field can be set in any order to generate the interrupt request. When the interrupt is serviced, the interrupt service routine may read PORTFE and logically AND it with PFEER to determine which pin(s) caused the interrupt. Once set, the edge-detect flag bits remains set, regardless of the subsequent state of the corresponding pin or changes in PAR programming, until the bit is cleared by software, or reset. To clear an edge-detect flag, the bit must be first read as a one and then the bit can be written to zero. Flags which are zero when the register is read are unaffected by the write operation. Also, if the edge detect logic detects another edge after the flag was read as a one and before a zero is written to clear it, the flag cannot be cleared until the flag is read as a one again and written to a zero. Bits in this register are set by the BIM and cannot be written to a one by software.

3.2.8.6 Port F Edge-Detect I/O Interrupt Enable (PFEER)

PFEER — Port F Edge-Detect I/O Register **0xYF FA3A**

MSB 7	6	5	4	3	2	1	LSB 0
PFEE7	PFEE6	PFEE5	PFEE4	PFEE3	PFEE2	PFEE1	PFEE0
RESET:							
0	0	0	0	0	0	0	0

Table 3-41 PFEER Bit Descriptions

Bit(s)	Name	Description
7:0	PFEER [7:0]	Port F I/O interrupt enable. The bits in this register individually enable interrupts by the corresponding edge-detect I/O pins. If a PFEER bit is written to one and the corresponding bit in the PORTFE register is set (or later becomes set when the specified edge occurs on the corresponding pin), an interrupt request is generated at the interrupt priority level specified by the PFEL[2:0] field of the port F interrupt level register. When the interrupt is serviced, the interrupt service routine may read PORTFE and logically AND it with PFEER to determine which pin(s) caused the interrupt. If the PFEER bit is written to zero, any pending I/O interrupt request is disabled. Also when specified edge occurs on the corresponding pin, an interrupt request is not generated. However, the PORTFE bit is set to record the edge event.

3.2.8.7 Port F Interrupt Level Register (PFLVR)

PFLVR — Port F Interrupt Level Register **0xYF FA3C**

MSB 7	6	5	4	3	2	1	LSB 0
0	IRQXL[2:0]			0	PFEL[2:0]		
RESET:							
0	1	1	1	0	0	0	0

Table 3-42 PFLVR Bit Descriptions

Bit(s)	Name	Description
7	—	Reserved
6:4	IRQXL	Port F priority level assignment. When the $\overline{\text{IRQX}}$ pin is configured as a level or edge sensitive IRQ pin (PFPAR[15:14] = 00 or 11), the IRQXL field specifies the interrupt request level of the IRQX pin, or defines the pin to be the $\overline{\text{BERR}}$ input. If IRQXL is not equal to 000, the pin functions as a level-sensitive or edge-sensitive interrupt request, generating the programmed level interrupt request when asserted. If IRQXL = 000, the pin functions as the $\overline{\text{BERR}}$ pin. The IRQXL field may be changed at any time. However, if the programmed edge or level change occurs on $\overline{\text{IRQX}}$ when the IRQXL field is being changed, the priority level of the new interrupt request is unpredictable. If an interrupt request is pending at the current level programmed in IRQXL, changing the value of IRQXL has no affect on this pending interrupt. However, future interrupt events on the IRQXL pin are reported at the new interrupt request level. The IRQXL field comes out of reset containing the value 111, which configures the pin as a level 7 interrupt pin ($\overline{\text{IRQ7}}$). This field has no affect if the pin function is programmed to be PF[7] (PFPAR[15:14] = 10 or 01).
3	—	Reserved
2:0	PFEL	Port F edge-detect I/O interrupt level. The interrupt priority level for all port F edge-detect I/O interrupts is determined by the priority level specified in the PFEL field. PFEL is reset to 0x0, which disables interrupt requests for the I/O edge detect logic. PFEL can be programmed to a value 0x0 (interrupt disabled) through 0x7 (highest priority). The port F edge-detect interrupt has the lowest arbitration priority in the BIM, when other BIM sources of interrupts are arbitrating at the same priority level. The PFEL field controls the interrupt request level of any pending digital I/O interrupts, even if the PFPAR is later programmed to select the primary pin function. The interrupt request level of pending digital I/O interrupts can be changed at any time.

3.2.9 Port G/H Operation

In master mode, emulation mode, MFTM, and SLAM, Port G/H functions as external data bus pins D[15:0]. In single chip mode, port G/H functions as digital I/O; each pin can be individually configured as either input or output. These pin functions cannot be changed, as there is no pin assignment register.

When a port G/H pin is configured as D[15:0], the BIM will enable the 3-V output driver in the associated pad. When a port G/H pin is configured as digital I/O, the BIM will enable the 5-V output driver in the associated pad.

The port G/H registers are accessible via the IMB unless the BIM is in emulation mode.

Table 3-43 Port G/H Supported Pin Functions

Pin	Master	Single Chip	MFTM	SLAM	Emulation[1]
D[15:0] / PG[7:0], PH[7:0]	D[15:0]	PG[7:0], PH[7:0]	D[15:0]	D[15:0]	D[15:0]

NOTES:
1. Digital I/O pin function provided by port replacement unit.

3.2.9.1 Port G Data Direction Register (DDRG)

DDRG — Port G Data Direction Register **0xYF FA2C**

MSB 7	6	5	4	3	2	1	LSB 0
PG[7]	PG[6]	PG[5]	PG[4]	PG[3]	PG[2]	PG[1]	PG[0]
RESET:							
0	0	0	0	0	0	0	0

Table 3-44 DDRG Bit Descriptions

Bit(s)	Name	Description
7:0	PG[7:0]	Port G data direction. The bits in this register control the direction of the port G pin drivers when the pins are configured as digital I/O. When the port is not defined to be digital I/O, its corresponding data direction bit has no effect. 0 = Corresponding pin is configured as an input 1 = Corresponding pin is configured as an output

3.2.9.2 Port G Output Data Register (PORTG)

PORTG — Port G Output Data Register **0xYF FA28**

MSB 7	6	5	4	3	2	1	LSB 0
PG[7]	PG[6]	PG[5]	PG[4]	PG[3]	PG[2]	PG[1]	PG[0]
RESET:							
0	0	0	0	0	0	0	0

Table 3-45 PORTG Bit Descriptions

Bit(s)	Name	Description
7:0	PG[7:0]	Port G output data. The PORTG register is used to store the data to be driven on the port G output pins, if the port is configured as output. When read, the current value of the PORTG register, not the port G pins, is returned.

3.2.9.3 Port G Pin Data Register (PORTGP)

PORTGP — Port G Pin Data Register **0xYF FA2A**

MSB 7	6	5	4	3	2	1	LSB 0
D[15]/PG[7]	D[14]/PG[6]	D[13]/PG[5:]	D[12]/ PG[4]	D[11]/PG[3]	D[10]/PG[2]	D[9]/PG[1]	D[8]/ PG[0]
RESET: Not affected by RESET.							
0	0	0	0	0	0	0	0

Table 3-46 PORTGP Bit Descriptions

Bit(s)	Name	Description
7:0	—	Port G data input. When read, the PORTGP register reflects the current state of the port G pins. Writes to PORTGP have no effect.

3.2.9.4 Port H Data Direction Register (DDRH)

DDRH — Port H Data Direction Register **0xYF FA2C**

MSB 7	6	5	4	3	2	1	LSB 0
PH[7]	PH[6]	PH[5]	PH[4]	PH[3]	PH[2]	PH[1]	PH[0]
RESET:							
0	0	0	0	0	0	0	0

Table 3-47 DDRH Bit Descriptions

Bit(s)	Name	Description
7:0	PH[7:0]	Port H data direction. The bits in this register control the direction of the port H pin drivers when the pins are configured as I/O. When the port is not defined to be digital I/O, its corresponding data direction bit has no effect. 0 = Corresponding pin is configured as an input 1 = Corresponding pin is configured as an output

3.2.9.5 Port H Output Data Register (PORTH)

PORTH — Port H Output Data Register **0xYF FA28**

MSB 7	6	5	4	3	2	1	LSB 0
PH[7]	PH[6]	PH[5]	PH[4]	PH[3]	PH[2]	PH[1]	PH[0]
RESET:							
0	0	0	0	0	0	0	0

Table 3-48 PORTH Bit Descriptions

Bit(s)	Name	Description
7:0	PH[7:0]	Port H output data. PORTH is used to store the data to be driven on the port H output pins, if the port is configured as output. When read, the current value of the PORTH register, not the port H pins, is returned.

3.2.9.6 Port H Pin Data Register (PORTHP)

PORTHP — Port H Pin Data Register **0xYF FA2A**

MSB 7	6	5	4	3	2	1	LSB 0
D[7]/PH[7]	D[6]/PH[6]	D[5]/PH[5:]	D[4]/ PH[4]	D[3]/PH[3]	D[2]/PH[2]	D[1]/PH[1]	D[0]/ PH[0]
RESET: Not affected by RESET.							
0	0	0	0	0	0	0	0

Table 3-49 PORTHP Bit Descriptions

Bit(s)	Name	Description
7:0	—	Port H data input. When read, PORTHP reflects the current state of the port H pins. Writes to PORTHP have no effect.

3.2.10 Port K Operation

In master mode and MFTM, port K can be configured as burst control/handshake signals or as digital I/O. In single chip mode, all port K pins can be configured as burst control/handshake signals or digital I/O, but only digital I/O is supported. In SLAM, port K pins DTACK, BREQ, and BTACK are used as cycle control signals. The other pins can be configured as burst chip select signals, $\overline{\text{DTACK}}$, or digital I/O. In emulation mode the pins are configured as burst control/handshake signals and BCS functions.

Table 3-50 indicates the port K pin functions supported for each operation mode. Even when a pin function is not listed as supported for a particular mode, that pin function can still be selected with the port K pin assignment register. However, the pin activity is undefined for that case.

When a port K pin is configured as its primary function, the BIM will enable the 3-V output driver in the associated pad. When a port K pin is configured as digital I/O, the BIM will enable the 5-V output driver in the associated pad.

Table 3-50 Port K Supported Pin Functions

Pin	Master	Single Chip	MFTM	SLAM	Emulation[1]
BCLK / PK[7]	BCLK or PK[7]	PK[7]	BCLK or PK[7]	BCLK or PK[7]	BCLK
$\overline{\text{BWE}}$ / PK[6]	BWE or PK[6]	PK[6]	BWE or PK[6]	PK[6]	BWE
$\overline{\text{BOE}}$ / PK[5]	BOE or PK[5]	PK[5]	BOE or PK[5]	PK[5]	BOE
$\overline{\text{BAA}}$ / PK[4]	BAA or PK[4]	PK[4]	BAA or PK[4]	PK[4]	BAA
$\overline{\text{LBA}}$ / PK[3]	LBA or PK[3]	PK[3]	LBA or PK[3]	PK[3]	LBA
$\overline{\text{BREQ}}$ / PK[2]	BREQ or PK[2]	PK[2]	BREQ or PK[2]	$\overline{\text{BREQ}}$ (in)	BREQ
$\overline{\text{BTACK}}$ / PK[1]	BTACK or PK[1]	PK[1]	BTACK or PK[1]	BTACK (out) or PK[1]	BTACK
$\overline{\text{DTACK}}$ / PK[0]	DTACK or PK[0]	PK[0]	DTACK or PK[0]	$\overline{\text{DTACK}}$ (out) or PK[0]	DTACK

NOTES:
1. Digital I/O pin function provided by port replacement unit

3.2.10.1 Port K Pin Assignment Register (PKPAR)

PKPAR — Port K Pin Assignment Register **0xYF FA26**

MSB 7	6	5	4	3	2	1	LSB 0
RESERVED				Burst CS/PK[7:3]	$\overline{\text{BREQ}}$, PK[2]	$\overline{\text{BTACK}}$,/PK[1] PK[1]	$\overline{\text{DTACK}}$/ PK[0]
RESET: See Table 3-3.							
0	0	0	0	0	0	0	0

Table 3-51 PKPAR Bit Descriptions

Bit(s)	Name	Description
7:4	—	Reserved
3	Burst CS/PK[7:3]	Port K pin configuration. This bit configures the port K[7:3] pins as either burst chip select signals (BCLK, $\overline{\text{BWE}}$, $\overline{\text{BOE}}$, $\overline{\text{BAA}}$ and $\overline{\text{LBA}}$) or digital I/O. 0 = Port K[7:3] is configured as digital I/O 1 = Port K[7:3] is configured as the burst chip select signals
2:1	—	Port K pin configuration. These bits configure the port K[2:1] pins as either burst mode handshake signals $\overline{\text{BREQ}}$ and $\overline{\text{BTACK}}$, or digital I/O. 0 = Port K[2:1] is configured as digital I/O 1 = port K[2:1] is configured as $\overline{\text{BREQ}}$ and $\overline{\text{BTACK}}$
0	$\overline{\text{DTACK}}$/PK[0]	Port K pin configuration. This bit configures the port K[0] pin as either external bus handshake signal $\overline{\text{DTACK}}$ or digital I/O. When this bit is cleared, port K[0] is configured as digital I/O. When set, port K[0] is configured as $\overline{\text{DTACK}}$. 0 = Port K[0] is configured as digital I/O 1 = port K[0] is configured as $\overline{\text{DTACK}}$

3.2.10.2 Port K Data Direction Register (DDRK)

DDRK — Port K Data Direction Register **0xYF FA24**

MSB 7	6	5	4	3	2	1	LSB 0
PK[7]	PK[6]	PK[5]	PK[4]	PK[3]	PK[2]	PK[1]	PK[0]
RESET:							
0	0	0	0	0	0	0	0

Table 3-52 DDRK Bit Descriptions

Bit(s)	Name	Description
7:0	PK[7:0]	Port K data direction. The bits in this register control the direction of the port K pin drivers when the pins are configured as I/O. When the port is not defined to be digital I/O, its corresponding data direction bit has no effect. 0 = Corresponding pin is configured as an input 1 = Corresponding pin is configured as an output

3.2.10.3 Port K Output Data Register (PORTK)

PORTK — Port K Output Data Register **0xYF FA20**

MSB 7	6	5	4	3	2	1	LSB 0
PK[7]	PK[6]	PK[5]	PK[4]	PK[3]	PK[2]	PK[1]	PK[0]
RESET:							
0	0	0	0	0	0	0	0

Table 3-53 PORTK Bit Descriptions

Bit(s)	Name	Description
7:0	PK[7:0]	Port K I/O data. PORTK is used to store the data to be driven on the port K output pins, when the port is configured as outputs. When read, the current value of the PORTK register, not the port K pins, is returned.

3.2.10.4 Port K Pin Data Register (PORTKP)

PORTKP — Port K Pin Data Register **0xYF FA22**

MSB 7	6	5	4	3	2	1	LSB 0
BCLK/ PK[7]	$\overline{\text{BWE}}$/ PK[6]	$\overline{\text{BOE}}$/ PH[5:]	$\overline{\text{BAA}}$/ PH[4]	$\overline{\text{LBA}}$/ PH[3]	$\overline{\text{BREQ}}$/PH[2]	$\overline{\text{BTACK}}$/PH[1]	$\overline{\text{DTACK}}$/ PH[0]
RESET: Not affected by RESET.							
0	0	0	0	0	0	0	0

Table 3-54 PORTKP Bit Descriptions

Bit(s)	Name	Description
7:0	—	Port K data input. When read, PORTKP reflects the current state of the port K pins. Writes to PORTKP have no effect.

3.3 Signal Definitions

This section contains a brief description of the input and output signals by their functional groups. Each signal is explained in a brief paragraph with reference (if applicable) to other sections that contain more detail about the function being performed. Specific timing information for each signal can be found in **APPENDIX E ELECTRICAL AND AC CHARACTERISTICS.**

3.3.1 Port A, B, and C Signals

In addition to the functions stated below, the following pins can also be configured to be discrete I/O pins. Refer to **3.2.4 Port A/B Operation** and **3.2.5 Port C Operation** for port control information.

3.3.1.1 Address Bus (A[23:0])

These three-state outputs provide the address for a bus transfer during all currently defined cycles, except CPU-space references. During CPU-space references, the address bus provides CPU related information. The address bus is capable of addressing 16 Mbytes (2^{24}) of data.

3.3.2 Port D Signals

In addition to the functions stated below, the following pins can also be configured to be discrete I/O pins. Refer to **3.2.6 Port D Operation** for port control information.

3.3.2.1 Chip Select ($\overline{CS}$[3:1])

These output signals select external devices at programmed addresses. Refer to **3.6 Asynchronous Chip Select (ACS)** for more information.

3.3.2.2 Freeze (FREEZE)

This output signal indicates that the CPU has acknowledged a breakpoint and has initiated background mode operation.

3.3.2.3 Quotient Out (QUOT)

This output furnishes either the quotient bit of the MSRA or the modulo counter clock. Refer to **3.4.6 Clock Control Registers** for more details on QUOT.

3.3.2.4 Emulation Mode Chip Selects (CSE1)

This output signal, along with CSE0, provides development support hooks. Refer to **3.1.9 Emulator Support** for more details.

3.3.2.5 Breakpoint ($\overline{BKPT}$)

This input signal indicates a hardware breakpoint to the CPU.

3.3.2.6 Development Serial In, Out, Clock (DSI,DSO, DSCLK)

These signals provide serial communications for background debug modes.

3.3.2.7 Instruction Pipe (IPIPE[2:0])

These output signals track the movement of words through the CPU instruction pipeline.

3.3.3 Port E Signals

In addition to the functions stated below, the following pins can also be configured to be discrete I/O pins. Refer to **3.2.7.1 Port E Pin Assignment Register (PEPAR)** for port control information.

3.3.3.1 Function Codes

These three-state outputs identify the processor state and the address space of the current bus cycle, as defined in **Table 3-55**.

Table 3-55 Function Code Assignments

FC2	FC1	FC0	Cycle Type
0	0	0	Reserved (Motorola)
0	0	1	User data space
0	1	0	User program space
0	1	1	Reserved (Motorola)
1	0	0	Reserved (Motorola)
1	0	1	Supervisor data space
1	1	0	Supervisor program space
1	1	1	Supervisor CPU-space operation

3.3.3.2 Chip Selects ($\overline{CS}$[7:5])

These output signals select external devices at programmed addresses. Refer to **3.6 Asynchronous Chip Select (ACS)** for information.

3.3.3.3 System Clock (CLKOUT)

This output signal is the MCU internal system clock. CLKOUT should be used as the bus cycle timing reference by external devices. This clock normally runs at the system clock speed but can be turned off in low power stop mode.

3.3.3.4 Transfer Size (SIZE)

This three-state output signal indicates the number of operand bytes to be transferred on the current bus cycle.

3.3.3.5 Address Strobe ($\overline{AS}$)

This three-state output is driven by the bus master to indicate that valid address, function codes, size, and R/$\overline{W}$ state information is on the bus.

3.3.3.6 Data Strobe ($\overline{DS}$)

In a read cycle, this three-state output signal is driven by the bus master to indicate that an external device should place valid data on the data bus. In a write cycle, data strobe indicates that the master device has placed valid data on the data bus.

3.3.3.7 Read/Write (R/$\overline{W}$)

This three-state output is driven by the bus master to indicate the direction of the data transfer on the bus. A logic one indicates a read from a slave device and a logic zero indicates a write to a slave device.

3.3.4 PORT F SIGNALS

In addition to the functions stated below, the following pins can also be configured to be discrete I/O pins. Refer to **3.2.8 Port F Operation** for port control information.

3.3.4.1 Interrupt Request ($\overline{IRQ}$[6:1], $\overline{IRQX}$)

These three-state inputs are prioritized input lines where $\overline{IRQ7}$ is the highest priority. These signals can be configured as level sensitive or edge sensitive.

$\overline{IRQ}$[6:1] and $\overline{IRQX}$ are internally maskable interrupts. $\overline{IRQX}$ can be configured to any interrupt request level. At reset, $\overline{IRQX}$ is configured as $\overline{IRQ7}$. Refer to **3.2.8.7 Port F Interrupt Level Register (PFLVR)** for more details.

3.3.4.2 Bus Error ($\overline{BERR}$)

This input signal indicates that an invalid bus operation is being attempted.

3.3.4.3 Chip Select ([$\overline{CS4}$])

This output signal selects external devices at programmable addresses. Refer to **3.6 Asynchronous Chip Select (ACS)** for information.

3.3.4.4 PORT G/H SIGNALS

In addition to the functions stated in this section, the following pins can also be configured to be discrete I/O pins. Refer to **3.2.9 Port G/H Operation** for port control information.

3.3.4.5 Data Bus (D[15:0])

These three-state bi-directional signals provide the general purpose data path between the MCU and all other devices.

3.3.5 Port K Signals

In addition to the functions stated below, the following pins can also be configured to be discrete I/O pins. Refer to **3.2.10 Port K Operation** for port control information.

3.3.5.1 Burst Chip Select Control (BCLK, BWE, BOE, BAA, LBA)

These signals implement the burst protocol. Refer to **3.6 Asynchronous Chip Select (ACS)** for more information.

3.3.5.2 Burst Request (BREQ)

This output signal indicates that a burst data transfer is requested by the CPU.

3.3.5.3 Burst Transfer Acknowledge (BTACK)

This input signal indicates that a word burst data transfer is complete. During a read cycle, data is latched when the processor recognizes $\overline{\text{BTACK}}$. Then another burst cycle is run, if needed; during a write cycle, when the processor recognizes $\overline{\text{BTACK}}$, the CPU continues with another burst cycle, if needed.

3.3.5.4 Data Transfer Acknowledge ($\overline{\text{DTACK}}$)

This input signal indicates that a data transfer is complete. During a read cycle, data is unlatched when the processor recognizes $\overline{\text{DTACK}}$ and the bus cycle terminates; during a write cycle the bus cycle is terminated when the processor recognizes $\overline{\text{DTACK}}$.

3.3.6 RESET OUT ($\overline{\text{RSTOUT}}$)

This output signal is an indication that the internal reset controller has reset the chip. When $\overline{\text{RESET}}$ and $\overline{\text{RSTOUT}}$ are both active, the user may drive his override options on the data bus.

3.3.6.1 Emulation Mode Chip Selects (CSE0)

This output signal, along with CSE1, provides development support signals. Refer to **3.1.9 Emulator Support** for more details.

3.3.7 RESET ($\overline{\text{RESET}}$)

This bi-directional open-drain signal is the system reset signal. The signal may be asserted internally or externally. Causes of assertion are power on reset, external reset, system protection reset, CPU reset instruction, test module reset, or loss of clock reset. In all cases, the $\overline{\text{RESET}}$ pin is guaranteed to be asserted by internal logic for a minimum of 512 clock cycles. Refer to **3.4 Clocks** for a complete description.

3.3.8 Crystal Oscillator (EXTAL, XTAL)

These two pins are the connections for an external crystal to the internal oscillator circuit. The EXTAL pin is used as the input in external clock mode. The clock source is configured during reset. Refer to **3.4 Clocks** for more details.

3.3.9 External Filter Capacitor (XFCN, XFCP)

These pins are used to add an external filter circuit for the phase locked loop. Refer to **3.4 Clocks** for more details.

3.3.10 Synthesizer Power (V_{DDSYN})

This pin provides separate power to the frequency synthesizer circuit. It is also used to configure system clock options during reset.

3.3.11 Synthesizer Ground (V_{SSSYN})

This pin is used to provide separate ground to the frequency synthesizer circuit.

3.3.12 External Bus Request ($\overline{EBR}$)

This input signal is used to prevent the MCU from using the external bus.

3.3.12.1 Three-State Control ($\overline{TSC}$)

When $\overline{EBR}$ is asserted simultaneously with the $\overline{RESET}$ pin, the MCU places all the MCU output drivers in a high-impedance state, except $\overline{RSTOUT}$ and $\overline{EBR}$.

3.3.12.2 Floating External Pins

All BIM I/O pins which are configured as inputs, must always be driven by external logic or pulled-up (or down) to avoid unnecessary current drain. The external bus pins, when they are configured for their primary function, will float under certain conditions as summarized in **Table 3-56**. Port G and H pins when configured as data bus pins may also float.

Table 3-56 Driven State of BIM External Bus Pins

Mode	A[23:16], A[15:0], FC[2:0], $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$, SIZE	$\overline{DTACK}$, $\overline{BERR}$	D[15:0]
Master mode — $\overline{EBR}$ negated	Driven by BIM	Driven by external logic	Driven by BIM in write cycles. Driven by external device in read cycles. Will float otherwise.
Master mode — $\overline{EBR}$ asserted	Pins not driven by BIM. Should be driven by external master.	Driven by external logic	Pins will float if not driven by external master.
Slave access mode	Driven by external logic	$\overline{DTACK}$ driven by BIM (only if $\overline{AS}$ asserted). $\overline{BERR}$ driven by external logic.	Driven by BIM in read cycles. Driven by external device in write cycles. Will float otherwise.
Single chip	Pin function not supported. Only function available is digital I/O.	Pin function not supported. Only function available is digital I/O.	Pin function not supported. Only function available is digital I/O.
In the cases where the pins will float, the user should drive the pins to either logic 0 or 1, or pull the pins up or down with external resistors for minimum power consumption.			

3.4 Clocks

The clock submodule consists of the crystal oscillator reference (OSC), the phase lock loop (PLL), the reduced frequency divider (RFD), the synthesizer control register

(SYNCR), the synthesizer status register (SYNST), and the clock control logic. shows the overall block diagram for the BIM/IMB clock generation.

To improve noise immunity, the PLL has its own set of power supply pins: V_{DDSYN} and V_{SSSYN}. All other circuits are powered by the normal internal supply pins, which are V_{DDI} and V_{SSI}.

The clock submodule also contains the reset control logic and the reset status register. See **3.4.8 RESET Operation** for details.

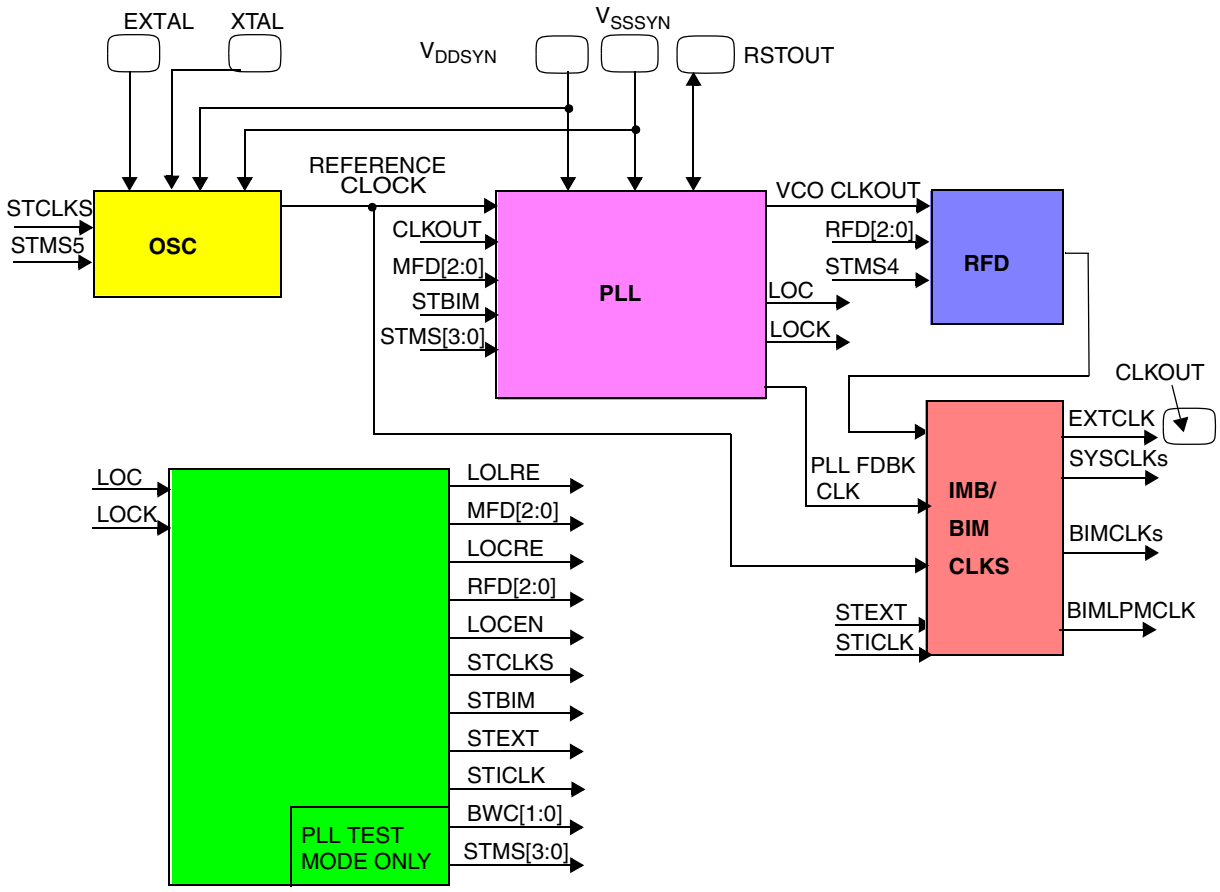


Figure 3-8 BIM Clocks Block Diagram

3.4.1 BIM Clock Modes

As shown in **Table 3-57**, the BIM clock mode is determined during reset by the state of the V_{DDSYN} pin and PCON[4] bit. The $\overline{LBA}$ pin is the override for PCON[4].

The value of V_{DDSYN} is latched during reset, but it is advisable to keep V_{DDSYN} at its reset level after reset is negated. If the V_{DDSYN} is changed during any reset other than power-on reset, the internal clocks may glitch as the clock source is changed between external clock mode and a PLL clock mode. This will affect the accuracy of the system

protection submodule timers. Whenever V_{DDSYN} is changed in reset, an immediate loss of lock condition occurs.

Table 3-57 Clock Source Configuration

V_{DDSYN}	$\overline{LBA}$ (PCON[4])	PLL Options	PLL Enabled
1	1	Normal PLL mode: $Fsys = Fref \cdot \frac{MFD + 2}{2^{RFD}}$	Yes
1	0	1:1 PLL mode: $Fsys = Fref$	Yes
0	X	External clock mode: $Fsys = \frac{Fref}{2}$	No

NOTES:
F_{ref} = input reference frequency
F_{sys} = CLKOUT frequency
MFD ranges from 0 to 7, see **Table 3-58**.
RFD ranges from 0 to 7, see **Table 3-58**.

When the normal PLL clock mode is selected, the PLL is fully programmable. The multiplication factor divider (MFD) provides frequency multiplication of 2X to 9X, and the RFD can be used to reduce the PLL clock frequency without disturbing the PLL.

The MFD and RFD bits have no affect. The external clock is divided by two internally to produce the various system clocks. Refer to **APPENDIX E ELECTRICAL AND AC CHARACTERISTICS** for external clock input requirements.

3.4.2 System Clocks Generation

The BIM uses the reference or the PLL to generate various system clocks. The reference is the crystal oscillator in the normal PLL clock mode. The reference is the external clock in the external and 1:1 PLL clock modes.

System clocks include the EXTCLK, SYSCLKs, BIMCLKs, and the BIMLPMCLK. The clock for the external system, EXTCLK, is provided on the CLKOUT pin when the CLKOUT function is selected by the PEPAR. See **3.2.7.1 Port E Pin Assignment Register (PEPAR)** for more information. SYSCLKs consisting of ICLOCK, ICLOCKE, and ICLK2XE are generated, routed on the IMB3 and are used by all internal IMB3 modules, except the BIM.

NOTE

ICLK2XE operates at two times the ICLOCK frequency. The primary BIM clocks are BIMCLKs. However the system protection submodule's timers, and the reset and port F pin synchronizers are clocked by BIMLPMCLK.

3.4.2.1 Programming System Clocks Frequency

In normal PLL clock mode, the default PLL frequency is two times the reference frequency after reset. The PLL frequency multiples that can be selected in the normal PLL clock mode are shown in **Table 3-58**. The frequency multiple is determined by the RFD and MFD bits in the SYNCR.

When programming the PLL, the system designer must be sure not to violate the maximum system clocks frequency. See **APPENDIX E ELECTRICAL AND AC CHARACTERISTICS** for the maximum allowable frequency The following steps are recommended to accommodate the frequency overshoot that occurs when the MFD bits are changed.

1. Determine the appropriate value for the MFD and RFD fields in the synthesizer control register (SYNCR). Note that the amount of jitter in the system clocks can be minimized by selecting the maximum MFD factor that can be paired with an RFD factor to provide the desired frequency.
2. Write a value of RFD = RFD (from step one) + 1 to the RFD field of the SYNCR.
3. Write the value determined in step one for the MFD field to the SYNCR.
4. Monitor the synthesizer lock bit (LOCK) in the synthesizer status register (SYNST). When the PLL achieves lock, write the RFD value determined in step one to the RFD field of the SYNCR. This changes the system clocks frequency to the desired frequency.

Table 3-58 PLL Frequency Multiples of The Reference — Normal PLL Mode[1]

RFD[2:0]	MFD[2:0] =							
	000 (2X)	001 (3X)	010 (4X)*	011 (5X)	100 (6X)	101 (7X)	110 (8X)	111 (9X)
0 = 000 (÷1)	2	3	4	5	6	7	8	9
1 = 001 (÷2)[2]	1	1.5	2	2.5	3	3.5	4	4.5
2 = 010 (÷4)	0.5	0.75	1	1.25	1.5	1.75	2	2.25
3 = 011 (÷8)	0.25	0.375	0.5	0.625	0.75	0.875	1	1.125
4 = 100 (÷16)	0.125	0.188	0.25	0.313	0.375	0.4385	0.5	0.563
5 = 101 (÷32)	0.0625	0.094	0.125	0.156	0.188	0.218	0.25	0.281
6 = 110 (÷64)	0.031	0.047	0.063	0.078	0.094	0.109	0.125	0.141
7 = 111 (÷128)	0.016	0.023	0.031	0.039	0.047	0.055	0.062	0.070

NOTES:
1. Keep maximum system clock frequency below specified limit given in **APPENDIX E ELECTRICAL AND AC CHARACTERISTICS**.
2. Denotes default value out of reset.

3.4.3 PLL Lock Detection

The lock detect logic monitors the reference frequency and the PLL feedback frequency to determine when frequency lock has been achieved. Phase lock is inferred by the frequency relationship, but is not guaranteed. The PLL lock status is reflected in the LOCK status bit in the SYNST. A sticky lock status indication, LOCKS, is also provided.

The lock detect function utilizes two counters which are clocked by the reference and PLL feedback respectively. When the reference counter has counted N cycles, the feedback counter's count is compared. If the feedback counter has also counted N cycles, the process is repeated for N + K counts. If the two counters' counts still match, the lock criteria is relaxed by 1/2 and the system is alerted that the PLL has achieved frequency lock.

After lock has been detected, the lock circuitry continues to monitor the reference and feedback frequencies using the alternate count and compare process. If the counters do not match at any comparison time, then the LOCK status bit is cleared to indicate that the PLL has lost lock. At this point, the lock criteria is tightened and the lock detect process is repeated.

The alternate count sequences prevent false lock detects due to frequency aliasing while the PLL tries to lock. Alternating between a tight and relaxed lock criteria prevents the lock detect function from randomly toggling between locked and not locked status due to phase sensitivities. **Figure 3-9** illustrates the sequence for detecting locked and not locked conditions.

In external clock mode, the PLL cannot lock since the PLL is disabled.

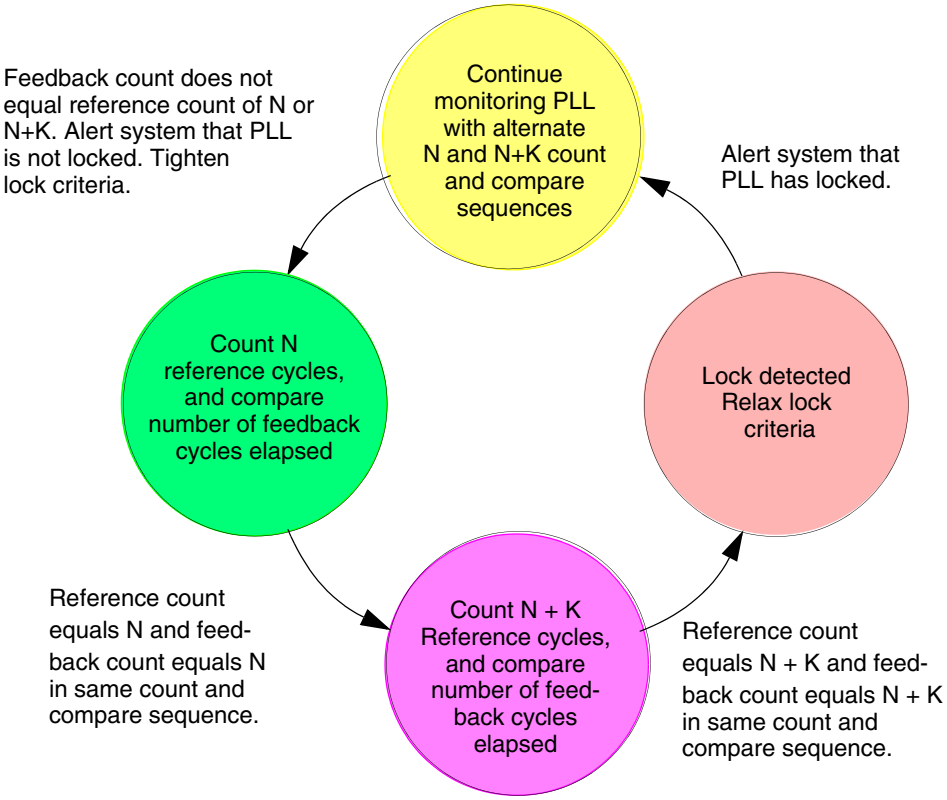


Figure 3-9 Lock Detect Sequence

3.4.3.1 PLL Loss of Lock Conditions

Once the PLL locks after reset, the LOCK and LOCKS status bits are set. If the MFD is changed in normal PLL mode or if an unexpected loss of lock condition occurs in normal or 1:1 PLL clock mode, the LOCK and LOCKS status bits are negated. The system clocks continue to be sourced from the PLL while the PLL attempts to relock.

NOTE

The PLL may overshoot when attempting to relock, causing the system clocks to exceed the maximum system frequency and possibly violating system timing characteristics.

Once locked, the LOCK bit is set. However the LOCKS bit remains cleared if the loss of lock was unexpected. The LOCKS bit is set to one when the loss of lock was caused by changing MFD.

3.4.3.2 PLL Loss of Lock RESET

The BIM provides the ability to assert reset when a loss of lock condition occurs by programming the LOLRE bit in the SYNCR. RESET is asserted if LOLRE is set. Since the LOCK and LOCKS bits in the SYNST are reinitialized after reset, the LOL bit must be read in the RSR to determine that a loss of lock condition occurred.

To exit reset in normal and 1:1 PLL modes, the reference must be present and the PLL must lock.

In external clock mode, the PLL cannot lock. Therefore a loss of lock condition cannot occur, and LOLRE has no affect.

3.4.3.3 Loss of Lock RESET during LPSTOP

The SYNCR can be programmed to disable the PLL during LPSTOP, causing a loss of lock. When the PLL is intentionally disabled in this manner, the loss of lock circuitry does not assert reset upon entering LPSTOP. See **Table 3-61** for when the PLL is disabled in LPSTOP mode.

3.4.4 Loss of Clock Detection

When enabled by the LOCEN bit in the SYNCR, the loss of clock (LOC) detection circuit monitors whether the system clocks are operating in normal or 1:1 PLL modes. When these clocks fail, this logic determines whether the failure is due to a PLL failure or a reference failure. In normal or 1:1 PLL modes, both the crystal oscillator and the PLL are monitored. In external clock mode, the loss of clock circuitry is disabled.

The LOC circuitry monitors both the reference and the PLL operation by monitoring the input clocks to the phase frequency detector shown in **Figure 3-10**. If the reference/feedback clock frequency falls below a minimum frequency, the LOC circuitry considers the clock to have failed and loss of clock status is reported by the LOCS bit in the SYNST. See **APPENDIX E ELECTRICAL AND AC CHARACTERISTICS** for the minimum clock frequency.

3.4.4.1 Alternate Clock Selection

Depending upon which clock source has failed, the LOC circuitry switches the system clocks source to the remaining operational clock. The system clocks are derived from the alternate clock source until reset is asserted. As shown in Table 3-59, if the reference fails in normal or 1:1 PLL modes, the PLL goes out of lock and into self-clocked mode (SCM). The PLL remains in SCM until the next reset. If the loss of clock condition is due to a PLL failure, the reference becomes the system clocks source until the next reset.

Table 3-59 Loss Of Clock Summary for Normal Operation

Clock Mode	System Clock Source Before Failure	Clock(s) Monitored by LOC Circuitry When LOCEN = 1	Reference Failure (Alternate Clock Selected by LOC Circuitry until RESET)	PLL Failure (Alternate Clock Selected by LOC Circuitry until RESET)
Normal PLL	PLL	Crystal oscillator, PLL	PLL self-clocked mode	Crystal oscillator
1:1 PLL	PLL	External clock, PLL	PLL self-clocked mode	External clock
External	External clock	None	None	NA

A special loss of clock condition occurs when both the reference and the PLL fail. The failures may be simultaneous or the PLL may fail first. In either case, the reference clock failure takes priority. The PLL attempts to operate in SCM. If successful, the PLL remains in SCM until the next reset. If the PLL cannot operate in SCM, the system remains static until the next reset. Both the reference and the PLL must be functioning properly to exit reset.

If the reference fails in external clock mode, there is no alternate clock to use as the system clocks source. The system remains static until the external clock begins operating again. The LOC condition is not reported, since the loss of clock circuitry is disabled.

3.4.4.2 Loss of Clock RESET

When a loss of clock condition is recognized, reset is asserted if the LOCRE bit in the SYNCR is set. The LOCS bit in the SYNST is cleared after reset, so the LOC bit must be read in the RSR to determine that a loss of clock condition occurred. LOCRE has no affect in external clock mode.

To exit reset in normal and 1:1 PLL modes, the reference must be present and the PLL must lock.

3.4.4.3 Loss of Clock During LPSTOP

SYNCR bits allow the PLL and/or reference clocks to be disabled during LPSTOP. When clocks are intentionally disabled in this manner, a loss of clock condition is not reported. RESET is not asserted upon entering LPSTOP when LOCRE is set. Instead, the loss of clock circuitry is adjusted to only monitor the clock(s) which remain(s) enabled. RESET is asserted if a monitored clock fails during LPSTOP when LOCRE

is set. The BIMCLK and SYSCLK will always match when the part is in LPSTOP except when the pll is disabled. For this case, the BIMCLK operates twice as fast as the SYSCLK. See Table 3-60 for more information.

Table 3-60 Loss of Clock Monitor Summary During LPSTOP

Clock Mode	STCLKS	STBIM	BIMLPMCLK /CLKOUT Source	Clock(s) Monitored by LOC Circuitry When LOCEN = 1	Reference Failure (Alternate Clock Used:)	PLL Failure (Alternate Clock Used:)
Normal PLL	0	0	Crystal oscillator	Crystal oscillator	PLL self-clocked mode	NA
1:1 PLL			External clock	External clock	PLL self-clocked mode	NA
External			External olock	None	None	NA
Normal PLL	0	1	PLL	Crystal oscillator, PLL	PLL self-clocked mode	Crystal oscillator
1:1 PLL			PLL	External clock, PLL	PLL self-clocked mode	External clock
External			External olock	None	None	NA
Normal PLL	1	x	Disabled	None	NA	NA
1:1 PLL			Disabled	None	NA	NA
External			Disabled	None	NA	NA

So long as the alternate clock remains functioning, LPSTOP can be exited via an interrupt. In normal and 1:1 PLL clock modes, the CPU is released from LPSTOP and execution continues with the alternate clock. Software should monitor LOCS to determine that a loss of clock condition occurred. RESET must be asserted to attempt to relock the PLL.

For external clock mode, the external clock must begin to operate before an interrupt or reset can be recognized to exit LPSTOP mode.

3.4.5 LPSTOP Operation

Low power stop (LPSTOP) mode is entered when the CPU performs a LPSTOP broadcast cycle after executing an LPSTOP instruction. In LPSTOP, some or all system clocks may be disabled to save power.

The STICLK, STCLKS, STBIM, STEXT bits in the SYNCR determine the affect of LPSTOP on the system clocks as follows:

- The BIMCLKs and SYSCLKs can be disabled via STICLK.
- The EXTCLK can be disabled via STEXT
- All system clocks can be disabled via STCLKS.
- The PLL is disabled via STBIM.

These LPSTOP options are summarized in Table 3-61.

RESET or any interrupt request that is higher than the interrupt mask priority level will exit LPSTOP mode. During LPSTOP the input synchronizers for interrupts and reset are clocked off the BIMLPMCLK clock, unless all clocks have been disabled. However when both the crystal oscillator and the PLL are disabled, reset operation and interrupt inputs are un-synchronized since no clock is active in the BIM. In this case, the only way to restart the system is by asserting reset or by asserting an external level interrupt request with higher priority than the LPSTOP interrupt priority mask level.

Table 3-61 LPSTOP Summary

STICLK	STCLKS	STBIM	STEXT	SYSCLKs and BIMCLKs Disabled	BIMLPMCLK Disabled	PLL Disabled	EXTOUT Disabled
0	0	0	0	No	No	Yes	No
0	0	0	1	No	No	Yes	Yes
0	0	1	0	No	No	No	No
0	0	1	1	No	No	No	Yes
1	0	0	0	Yes	No	Yes	No
1	0	0	1	Yes	No	Yes	Yes
1	0	1	0	Yes	No	No	No
1	0	1	1	Yes	No	No	Yes
X	1	X	X	Yes	Yes	Yes	Yes

3.4.5.1 Incurred Clock Start-Up

When the PLL is disabled in LPSTOP mode, exiting LPSTOP via an interrupt requires the PLL to re-lock for normal or 1:1 PLL clock modes. System clocks are derived from the reference until the PLL has locked. Then the CPU is released from LPSTOP mode and the system clocks are sourced from the PLL.

NOTE

When EXTCLK is driven during LPSTOP, the source of EXTCLK is switched on-the-fly from the reference to the PLL after it locks.

If the crystal oscillator is disabled during LPSTOP, exiting LPSTOP via an interrupt requires the normal crystal start up time plus PLL lock time for normal or 1:1 PLL clock modes. Until the crystal oscillator starts, the PLL operates in self-clocked mode. When a clock is recognized on the reference input, the PLL starts the locking process. After the PLL locks, the CPU is released from LPSTOP mode and the system clocks are sourced from the PLL.

NOTE

EXTCLK is not driven until the PLL locks.

There is no start-up time when the PLL remains active during LPSTOP.

Whenever LPSTOP is exited via reset, the reference must be present and the PLL must lock before execution can continue in normal and 1:1 PLL modes.

In external clock mode, the reference must be present to exit LPSTOP via reset or via an interrupt.

3.4.6 Clock Control Registers

The clock operation is controlled by the synthesizer control register (SYNCR) and status is reported in the synthesizer status register (SYNST). The following sections describe these registers in detail.

3.4.6.1 Synthesizer Control Register (SYNCR)

The synthesizer control register (SYNCR) is readable and writable in supervisor mode. It contains bits for defining the clock operation for the system. Reserved bits are not affected by writes and always return zero on a read. SYNCR register shows the definition of this register. Following the register definition is an expanded description of each bit field.

SYNCR — Synthesizer Control Register **0xYF FA08**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
LOLRE	MFD[2:0]			LOCRE	RFD[2:0]			STI-CLK	ST-CLKS	STBIM	STEXT	LO-CEN	RESERVED		
RESET:															
1	0	1	0	1	0	0	1	0		0	0	1	0	0	0

Table 3-62 SYNCR Bit Descriptions

Bit(s)	Name	Description
15	LOLRE	Loss of lock reset enable. When the PLL is operating in normal or 1:1 PLL mode, the PLL must be locked before setting the LOLRE bit. Otherwise reset is immediately asserted. The PLL is disabled in external clock mode, so the PLL does not lock. Since LOLRE is intended to handle a loss of lock, the LOLRE bit has no affect in external clock mode. LOCEN does not affect the LOLRE operation. See **3.4.3 PLL Lock Detection** for additional information. 0 = RESET is not asserted when a loss of lock indication occurs 1 = RESET is asserted when a loss of lock indication occurs
14:12	MFD	Multiplication factor divider. The MFD bits control the value of the divider in the PLL feedback loop. The value specified by the MFD bits establish the multiplication factor applied to the reference frequency. The bit field encoding is shown in row one of **Table 3-58**. When the MFD bits are changed, the PLL loses lock. To prevent an immediate reset, the LOLRE bit must be cleared before writing the MFD bits. In 1:1 PLL mode, the MFD bits are ignored and the multiplication factor is set to 1X. In external clock mode the MFD bits have no affect.
11	LOCRE	Loss of clock reset enable. The LOCRE bit determines how the BIM handles a loss of clock condition. If the LOCS bit in the SYNST indicates a loss of clock condition, setting the LOCRE bit causes an immediate reset. Additionally, the LOC bit in the reset status register (RSR) should be checked before setting the LOCRE bit to avoid a hardware/software loop disabling the system. Since reset must be asserted to allow the PLL to relock after entering SCM, both LOCEN and LOCRE are enabled after reset. In external clock mode LOCRE has no affect, since the loss of clock circuitry is disabled. 0 = RESET is not asserted when a loss of lock indication occurs. Has no affect when LOCEN = 0 1 = RESET is asserted when a loss of lock indication occurs

Table 3-62 SYNCR Bit Descriptions (Continued)

Bit(s)	Name	Description
10:8	RFD	Reduced frequency divider. The RFD bits control a prescaler at the output of the PLL. The value specified by the RFD bits establish the divisor applied to the PLL frequency. The RFD bit field encoding is shown in column one in **Table 3-58**. Changing the RFD bits does not affect the PLL's VCO, (i.e., no re-lock delay is incurred.) Resulting changes in clock frequency are synchronized to the next falling edge of the current system clock. However these bits should only be written when the lock bit (LOCK) is set, to avoid surpassing the allowable system operating frequency. In external clock mode the RFD bits have no affect.
7	STICLK	Stop internal clocks. The STICLK bit determines whether the BIMCLKs and SYSCLKs continue to operate in LPSTOP mode. 0 = BIMCLKs and SYSCLKs are enabled 1 = BIMCLKs and SYSCLKs are disabled (normal LPSTOP operation)
6	STCLKS	Stop clocks. The STCLKS bit determines whether any clocks are operational in LPSTOP mode. See **Table 3-61**. 0 = Reference clock continues to operate normally 1 = Reference and PLL are disabled
5	STBIM	Stop BIM clocks. The STBIM bit determines the clock source for BIMLPMCLK in LPSTOP mode. If STCLKS = 1, the STBIM bit has no affect since all clocks are stopped. See **Table 3-61**. In external clock mode, this bit has no affect; the BIMLPMCLK clock source is the external clock. 0 = BIMLPMCLK clock source is the reference and the PLL is disabled 1 = BIMLPMCLK clock source is the PLL
4	STEXT	Stop BIM clocks. The STEXT bit determines whether EXTCLK is driven during LPSTOP mode. The clock source for EXTCLK is determined by STBIM. If STCLKS = 1, STEXT has no affect on EXTCLK since all clocks are disabled. See **Table 3-61**. EXTCLK may also be disabled at the CLKOUT pin by selecting digital I/O as the pin function. Refer to **3.2.7.1 Port E Pin Assignment Register (PEPAR)** for additional information. 0 = EXTCLK is driven 1 = EXTCLK is not driven
3	LOCEN	Loss of clock enable. The LOCEN bit determines whether the loss of clock function is operational. In external clock mode, this bit has no affect; the loss of clock circuitry is disabled. LOCEN does not affect the loss of lock circuitry. When a loss of lock occurs, both LOCK and LOCKS status bits are cleared. See **3.4.4 Loss of Clock Detection** for more information. EXTCLK may also be disabled at the CLKOUT pin by selecting digital I/O as the pin function. Refer to **3.2.7.1 Port E Pin Assignment Register (PEPAR)** for additional information. 0 = Loss of clock function is disabled 1 = Loss of clock function is enabled
2:0	—	Reserved

3.4.6.2 Synthesizer Status Register (SYNST)

The synthesizer status register (SYNST) is an 8-bit read-only register. Reserved bits return zero when read. The synthesizer status register shows the definition of this register. Following the register definition is an expanded description of each bit field.

SYNST — Synthesizer Status Register **0xYF FA0A**

MSB 15	14	13	12	11	10	9	LSB 8
MODE[1:0]		RESERVED			LOCKS	LOCK	LOCS
RESET:							
X		0	0	0	X	X	0

X indicates the bits are decided at reset based on the clock mode.

Table 3-63 SYNST Bit Descriptions

Bit(s)	Name	Description
15	MODE	Clock mode. The MODE bits are determined at reset. 00 = External clock 01 = External clock 10 = 1:1 PLL 11 = Normal PLL
13:11	—	Reserved
10	LOCKS	PLL lock status bit. The LOCKS bit is a sticky indication of PLL lock status. LOCKS is set by the lock detect circuitry when the PLL acquires lock after: 1) a system reset; 2) exiting LPSTOP mode; or 3) a write to the SYNCR which modifies the MFD bits in normal PLL mode. Whenever the PLL loses lock, LOCKS is cleared. LOCKS remains cleared even after the PLL relocks, until one of the three previously-stated conditions occurs. Furthermore, if the LOCKS bit is read when the PLL simultaneously loses lock, the bit does not reflect the current loss of lock condition. If operating in external clock mode, LOCKS remains cleared after reset. In normal and 1:1 PLL modes, LOCKS is set after reset.
9	LOCK	PLL lock status. The LOCK bit indicates whether the PLL has acquired lock. The frequency is specified by the MFD and RFD bits in the SYNCR register for normal PLL clock mode. The frequency is 1X the reference in 1:1: PLL clock mode. The PLL is disabled in external clock mode. The PLL loses lock when a frequency deviation of greater than approximately 1.5% occurs. PLL lock occurs when the synthesized frequency matches to within approximately 0.75% of the desired frequency. If the LOCK bit is read when the PLL simultaneously loses lock, the bit does not reflect the current loss of lock condition. The LOCK bit is used by the power-on-reset circuit as a condition for releasing reset. See **3.3.7 RESET (RESET)** for additional information. If operating in external clock mode, LOCK remains cleared after reset. In normal and 1:1 PLL modes, LOCK is set after reset. 0 = PLL is not yet locked or the PLL has lost lock 1 = PLL is locked
8	LOCS	Sticky loss of clock status. The LOCS bit is a sticky indication of whether a loss of clock condition has occurred at any time since exiting reset in normal and 1:1 PLL modes. If the read of the LOCS bit and the loss of clock condition occur simultaneously, the bit does not reflect the current loss of clock condition. A loss of clock condition can only be detected when LOCEN = 1. See **3.4.4 Loss of Clock Detection**. LOCS is always zero in external clock mode. 0 = System clocks are operating normally 1 = System clocks have failed due to a reference failure or a PLL failure

3.4.7 PLL Operation

In normal PLL mode, the system clocks are synthesized by a PLL that is capable of multiplying the reference clock frequency by 2X to 9X, provided the system clock (CLKOUT) frequency remains within the range listed in **APPENDIX E ELECTRICAL AND AC CHARACTERISTICS**. For example, if the reference frequency is one MHz, the PLL can synthesize frequencies of two MHz to nine MHz. In addition, the output of the PLL can be divided down to reduce the system frequency with the RFD. The RFD is not contained in the feedback loop of the PLL, so changing the RFD bits does not affect PLL operation. See **Table 3-62**.

NOTE

The system frequency programmed should not exceed the maximum system clock frequency allowed, refer to **APPENDIX E ELECTRICAL AND AC CHARACTERISTICS** for the PLL.

Figure 3-10 shows the overall block diagram for the PLL. Each of the major blocks shown is discussed briefly below.

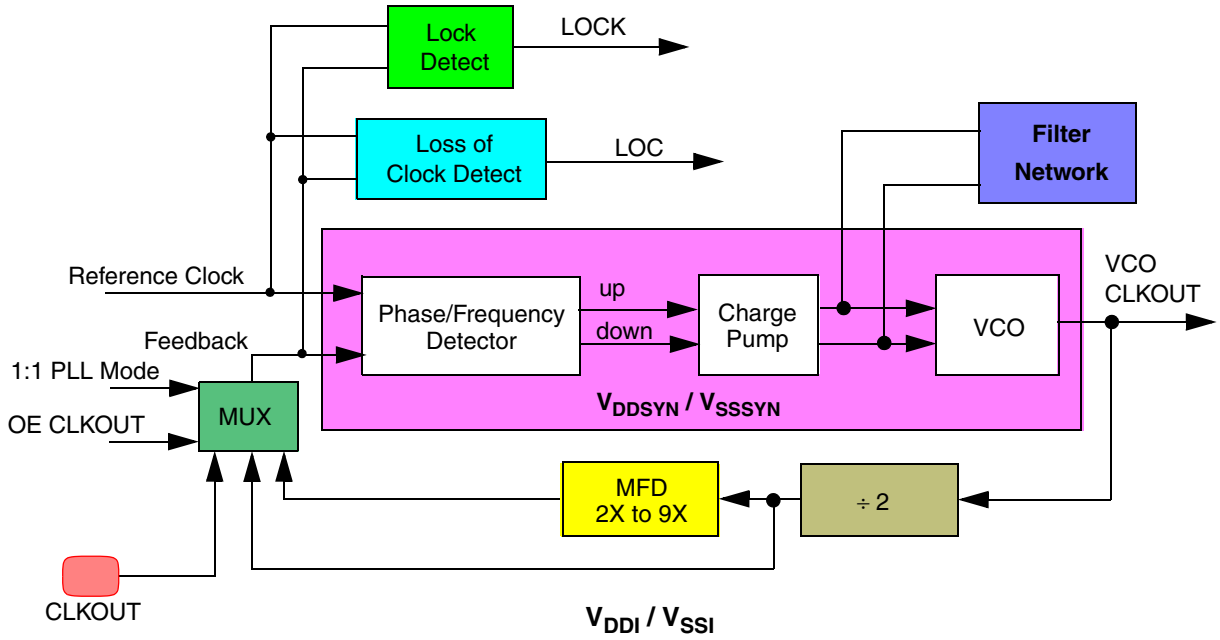
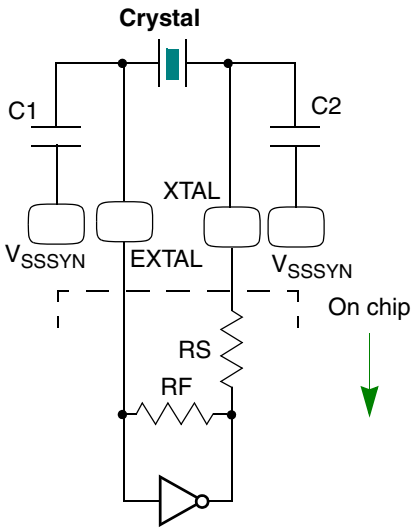


Figure 3-10 PLL Block Diagram Crystal Oscillator

The external support circuitry for the crystal oscillator is shown in Figure 3-11. Suggested component values are shown as well.

NOTE

The actual circuit should be reviewed with the crystal manufacturer.



8-MHz Crystal Configuration
C1 = C2 = 16 PF
RF = 1 MW
RS = 470 W

Figure 3-11 Crystal Oscillator

3.4.7.1 Phase/Frequency Detector (PFD)

The PFD is a quad-latch type IV phase-frequency detector. It compares both the phase and frequency of the reference clock and the feedback clock. The reference clock comes from either the crystal oscillator or an external clock source. The feedback clock comes from either CLKOUT in 1:1 PLL mode, two divide by two blocks if CLKOUT is disabled in 1:1 PLL mode, or from the VCO output divided down by the MFD in normal PLL mode.

When the frequency of the feedback clock is less than half the reference clock, the PFD asserts the UP signal continuously. When the feedback frequency is less than the reference clock frequency but greater than half its frequency, than the UP signal will be pulsed for a fraction of each reference clock cycle. When the frequency of the feedback clock equals the frequency of the reference clock (i.e., the PLL is frequency locked), the PFD will pulse the UP or DOWN signals depending on the relative phase of the two clocks. If the falling edge of the feedback clock lags the falling edge of the reference clock, then the UP signal is pulsed. If the falling edge of the feedback clock leads the falling edge of the reference clock, then the DOWN signal is pulsed. The width of these pulses relative to the reference clock is dependent on how much the two clocks lead or lag each other. Once phase lock is achieved, the PFD continues to pulse the UP and DOWN signals for a very short duration during each reference clock cycle. These short pulses force the PLL to continually update and prevent a frequency drift phenomenon referred to as “dead-banding.”

3.4.7.2 Charge Pump/Loop Filter

The charge pump operates in two modes which are automatically selected during the phase locking process. Initially, the charge pump operates in its high current mode (PLL wide bandwidth mode) which enables the PLL frequency to ramp up quickly.

Then as frequency/phase lock is achieved, the charge pump switches to its low current mode (PLL narrow band mode) minimizing the level of systematic jitter in the PLL output clock.

Actual operation of the charge pump is controlled by the UP and DOWN signals from the PFD. They control whether the charge pump applies or removes charge, respectively, from the loop filter.

3.4.7.3 VCO

The voltage formed across the loop filter controls the frequency of the VCO output. The frequency to voltage relationship (or VCO gain) is positive and the output frequency is four times the target system frequency.

3.4.7.4 MFD

The MFD divides down the output of the VCO and feeds it back (when not in 1:1 PLL mode) to the PFD. The PFD controls the VCO frequency (via the charge pump and loop filter) such that the reference and feedback clocks have the same frequency and phase. Thus, the input to the MFD, which is also the output of the VCO, is at a frequency that is the reference frequency multiplied by the same amount that the MFD divides by. For example, if the MFD divides the VCO frequency by six, then the PLL will be frequency locked when the VCO frequency is six times the reference frequency. The presence of the MFD in the loop allows the PLL to perform frequency multiplication, or synthesis. **Table 3-58** shows the possible MF values. When the PLL is operating in 1:1 PLL mode, the MFD is bypassed and the effective multiplication factor is 1X. See **Table 3-57**.

3.4.7.5 Clock Delay

In 1:1 PLL mode, however, the RFD is disabled. The feedback clock comes directly from the CLKOUT pin and true phase lock is achieved. However, if the CLKOUT pin is disabled while operating in 1:1 PLL mode the feedback input for the PLL will be derived from two divide by two blocks. The transition will cause a momentary phase error between the reference and the internal system clock. The reverse situation of enabling CLKOUT while operating in 1:1 PLL mode with CLKOUT disabled will also cause a momentary phase error.

3.4.8 RESET Operation

The BIM has several sources of reset as shown in **Figure 3-12**. The purpose of the reset control logic is to determine the cause of reset, synchronize reset if necessary to the completion of the current bus cycle, and assert the appropriate reset lines. Each reset source has a status bit associated with it and is explained in detail in **3.4.8.13 RESET Status Register (RSR)**.

The pin configuration during reset is described in **3.1.5 RESET Configuration**. To avoid conflicts on the configuration pins, the reset values of the configuration pins should be driven only while the $\overline{\text{RESET}}$ pin and the $\overline{\text{RSTOUT}}$ pin are low. Note that the $\overline{\text{RSTOUT}}$ pin is multi-functional. When the BIM is in emulation mode, $\overline{\text{RSTOUT}}$ is mul-

tiplexed with the CSE0 pin function. While the RESET pin is low, the state of the BIM internal reset signal, SRESET, is driven by the BIM on the RSTOUT pin. Refer to the **APPENDIX E ELECTRICAL AND AC CHARACTERISTICS** for timing information.

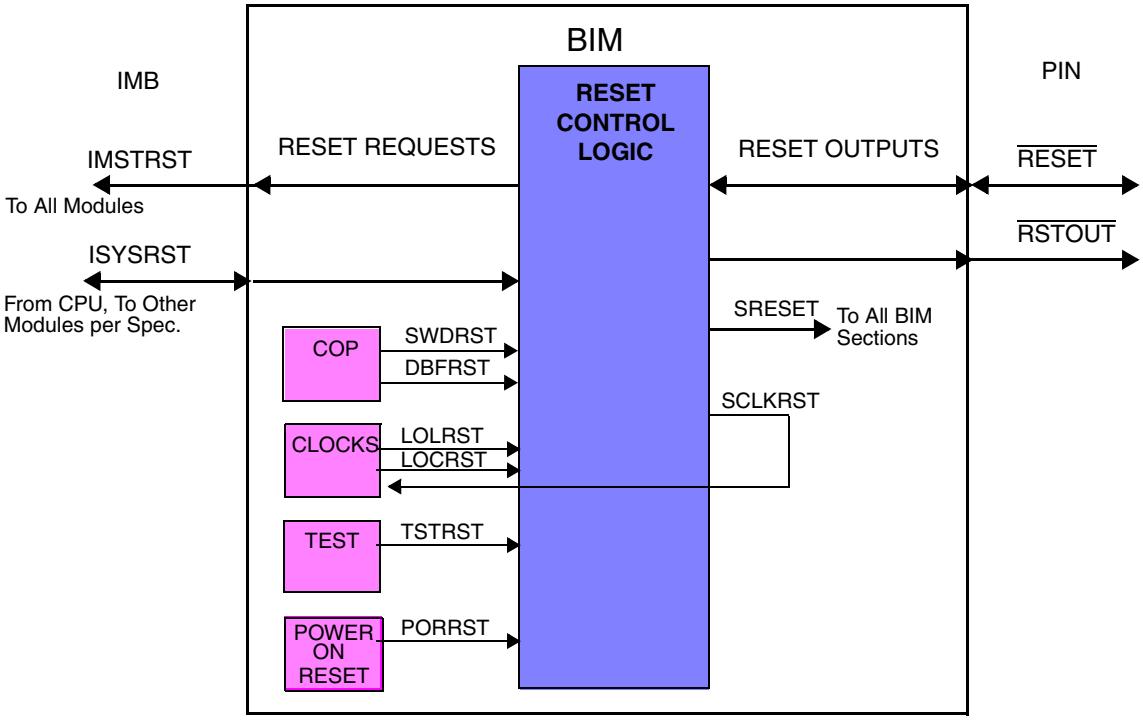


Figure 3-12 RESET Block Diagram

3.4.8.1 Sources of RESET

Table 3-64 defines the sources of reset and the signals driven by the BIM reset controller. External reset, test submodule reset, and loss-of-clock reset are synchronous master reset sources. Asynchronous master reset sources are power-on reset, software watchdog, double bus fault, and loss of lock. The RESET instruction is a system reset source only.

Synchronous master reset sources are not acted upon by the reset control logic until the end of the current bus cycle to protect data integrity. Whenever the reset control logic must synchronize reset to the end of the bus cycle, the internal bus monitor is automatically enabled regardless of the BME bit setting in the SYPCR. Then if the current bus cycle is not terminated normally, the bus monitor eventually terminates the cycle based on the length of time programmed in the BMT field of the system protection control register.

Single-byte or aligned-word writes on the IMB are guaranteed to complete without data corruption when a synchronous reset occurs. External writes are also guaranteed to complete, provided the external configuration logic on the configuration pins is condi-

tioned by the RSTOUT pin being low. A long word write, a misaligned operand write, or a read cycle are not guaranteed to complete.

Asynchronous master reset sources usually indicate a catastrophic failure. The reset control logic does not wait for the current bus cycle to complete. RESET is asserted immediately to the system. **Table 3-64** gives a summary of the action that occurs for each reset source.

Table 3-64 RESET Source Summary

Source	Timing	RESET Lines Asserted by RESET Controller				
External RESET pin	synch	IMSTRST[1]	SRESET[2]	SCLKRST[3]	SMCRSTB[4]	ISYSRST[5]
Power-on	async	IMSTRST	SRESET	SCLKRST	SMCRSTB	—
Sys prot WDOG	async	IMSTRST	SRESET	SCLKRST	SMCRSTB	ISYSRST
Sys prot DBF	async	IMSTRST	SRESET	SCLKRST	SMCRSTB	ISYSRST
Loss of clock	synch	IMSTRST	SRESET	SCLKRST	SMCRSTB	ISYSRST
Loss of lock	async	IMSTRST	SRESET	SCLKRST	SMCRSTB	ISYSRST
Test	synch	IMSTRST	SRESET	—	SMCRSTB	ISYSRST
CPU RESET instruction	synch	—	—	—	SMCRSTB	—

NOTES:
1. IMSTRSTB is the master reset line that goes to all modules on the IMB.
2. SRESET is the reset line that goes to all other internal circuits in the BIM.
3. SCLKRST is the reset line that resets the clock sub-module.
4. SMCRSTB is the reset line that drives the external reset pin.
5. ISYSRSTB is the system reset line that goes to all modules on the IMB. The reset controller asserts ISYSRSTB for one clock tic before asserting IMSTRSTB, except for POR.

3.4.8.2 External RESET

Since the RESET pin is a bi-directional line, a conflict exists when the CPU executes a RESET instruction and when an external device asserts the RESET pin. To guarantee that an external reset will be recognized by the EBI, the RESET pin must be held for at least 750 CLKOUT cycles so that it overlaps the 512 cycles of an internal reset.

If the CPU RESET instruction is not used, an external RESET assertion as short as four cycles will be recognized and latched by the BIM. The BIM then asserts RESET for approximately 512 cycles.

3.4.8.3 Power-On RESET

Upon power up, the reset controller asserts RESET and maintains the assertion of RESET for approximately 512 cycles after V_{DD} has reached a minimum acceptable level. Refer to **APPENDIX E ELECTRICAL AND AC CHARACTERISTICS**. Additionally, when 1:1 PLL clock mode or normal PLL clock mode is selected, RESET remains asserted until the PLL achieves phase-lock. The power on reset signal is now driven down the IMB.

3.4.8.4 Software Watchdog RESET

This reset condition occurs when the IRSEL bit in the SYPCR is cleared and the software watchdog counts down to zero. The reset controller asserts RESET for approximately 512 cycles, and then exits reset and resumes operation.

3.4.8.5 Double Bus Fault RESET

This reset condition occurs when the DBE bit in the SYPCR is set and HALT is asserted after a double bus fault. The reset controller asserts RESET for approximately 512 cycles, and then exits reset and resumes operation.

3.4.8.6 Loss of Clock RESET

This reset condition occurs when the reference or PLL fails in 1:1 or normal PLL mode and the LOCRE bit in the SYNCR register is set. The reset controller asserts RESET for approximately 512 cycles, and then exits reset and resumes operation.

3.4.8.7 Loss of Lock RESET

This reset condition occurs when the PLL loses lock and the LOLRE bit in the SYNCR register is set. The reset controller holds RESET for approximately 512 cycles, and then exits reset and resumes operation.

3.4.8.8 Test RESET

This reset condition occurs when the BIM is in test mode and the test module requests this reset as a result of its completion of an automatic scan operation. The reset controller asserts RESET for approximately 512 cycles, and then exits reset and resumes operation.

3.4.8.9 System RESET

A system reset occurs when the CPU executes a RESET instruction. The RESET instruction allows software to reset the system to a known state and then continue processing with the next instruction. The RESET instruction does not cause loading of the reset vector or affect any internal CPU registers or BIM configuration registers. The RESET instruction does assert the external RESET pin to reset external devices. The CPU drives ISYSRST, which may be used by modules other than the BIM. It is the responsibility of the CPU to assert reset at the proper time in the bus cycle, and to maintain reset in the asserted state for approximately 512 cycles (C32 X 256 cycles). Not all CPUs support driving ISYSRST.

The reset controller uses the ISYSRST input only as an indication to drive the SMCRSTB signal. RSTOUT will not be asserted for a system reset since the SRESET signal is not asserted, (i.e., the BIM is not reset).

After ISYSRST is negated by the CPU, the reset controller waits approximately 10 clocks for the external pullup on the RESET pin to negate the pin, and then samples the pin. If the RESET pin is high, the reset controller releases the MCU to continue bus cycles. If the pin is still low, it waits approximately 180 further clocks and samples the

pin again. If the RESET pin is negated at this point, the EBI is released to run bus cycles. If the pin is still low, an external reset sequence is initiated.

3.4.8.10 RESET Assertion by an External Device

If the RESET pin is driven low by an external device for at least four CLKOUT cycles, the reset control logic will latch the reset request internally, but the MCU is not yet in reset (13). When the current bus cycle is completed, SCLKRST and IMSTRSTB are asserted. The MCU is now in reset(19). When the reset control logic detects that the RESET pin is not being driven low externally(10), the reset control logic will then drive the RESET pin low for an additional 512 cycles to guarantee that a minimum 512 cycle reset is generated to the entire system(3). At the end of 512 cycles, the reset control logic releases the RESET pin (high impedance)(5), and the external configuration on the appropriate pins is latched(4) (see **3.1.5 RESET Configuration**).

Now the reset control logic begins waiting for 10 additional clock cycles, to allow the external pullup device on RESET to negate the pin(5). At the end of this 10 cycle period, the reset logic samples the RESET pin(6). If the RESET pin is high, the control logic releases the EBI to begin running bus cycles(9). If the RESET pin is still asserted low, the control logic waits for 180 further clock cycles and samples the pin again(7). If the pin is high, the control logic releases the MCU to begin running bus cycles(9). If RESET is still low, the reset control logic will wait for the pin to go high(10), and then drive the RESET pin low again for 512 cycles(3). At the end of this 512 clock period, the reset configuration is re-latched(4), and the pin-sampling sequence is repeated.

3.4.8.11 Internal RESET Request

If reset is asserted due to a synchronous internal reset source, such as test module reset(14), the reset control logic will wait for a bus cycle termination(18), and then drive internal and external reset for 512 system clock cycles(18,3). If reset is asserted by an asynchronous reset source, such as double bus fault or software watchdog(11), the reset control logic drives internal and external reset for 512 system clock cycles without waiting for a bus cycle termination(3). Thus, any internal source of reset is guaranteed to cause at least a 512 cycle assertion of reset. After the 512 cycles, the reset control logic will re-latch the configuration and continue the previously described sequence.

3.4.8.12 Power-On RESET

When the reset sequence is initiated by power-on reset(1), the same reset sequence is followed as other asynchronous reset sources.

3.4.8.13 RESET Status Register (RSR)

The reset status register (RSR) contains a status bit for every reset source in the BIM. When the BIM enters reset, the cause(s) of the reset condition is/are latched along with a value of zero for the other reset sources that did not cause the reset condition. These values are then reported via the RSR; one or more status bits may be set. The cause of any subsequent reset is also recorded in the register, overwriting status from

the previous reset condition. If a system reset occurs simultaneously with any other type of reset, the RSR only records the master reset source.

This register can be read at any time. This register cannot be written.

RSR — RESET Status Register **0xYF FA0A**

MSB 7	6	5	4	3	2	1	LSB 0
EXT	POW	SW	DBF	LOL	LOC	SYS	—
RESET:							
0	0	0	0	0	0	0	0

Table 3-65 RSR Bit Descriptions

Bit(s)	Name	Description
7	EXT	External reset. This bit indicates that the last reset state was caused by an external device asserting the RESET pin.
6	POW	Power up reset. This bit indicates that the last reset state was caused by the power-up reset circuit, which is part of the reset controller.
5	SW	Software watchdog reset. This bit indicates that the last reset state was caused by the software watchdog circuit in the system protection module.
4	DBF	Double bus fault. This bit indicates that the last reset state was caused by a double bus fault as detected by the system protection module.
3	LOL	Loss of lock reset. This bit indicates that the last reset state was caused by a loss of lock as detected by the PLL circuits.
2	LOC	Loss of clock reset. This bit indicates that the last reset state was caused by a loss of clock as detected by the loss of clock circuitry.
1	SYS	System reset. This bit indicates that the last reset state was caused by the CPU executing a RESET instruction.
0	—	Reserved

3.5 System Protection

The burst integration module (BIM) provides features to safeguard MCU operations within different operating environments. Many functions normally implemented externally, whether by glue logic, or in software, have been integrated on-chip. The BIM includes bus cycle timeout monitors, a real-time clock which can operate in low power mode, and a software watchdog timer for runaway software protection. The three monitors in the system protection sub-module are: the bus monitor, the spurious interrupt monitor and the double bus fault monitor.

Details on the programming model for system protection registers can be found in **3.5.5 Programming Model**.

Figure 3-13 represents a graphical overview of the system protection section.

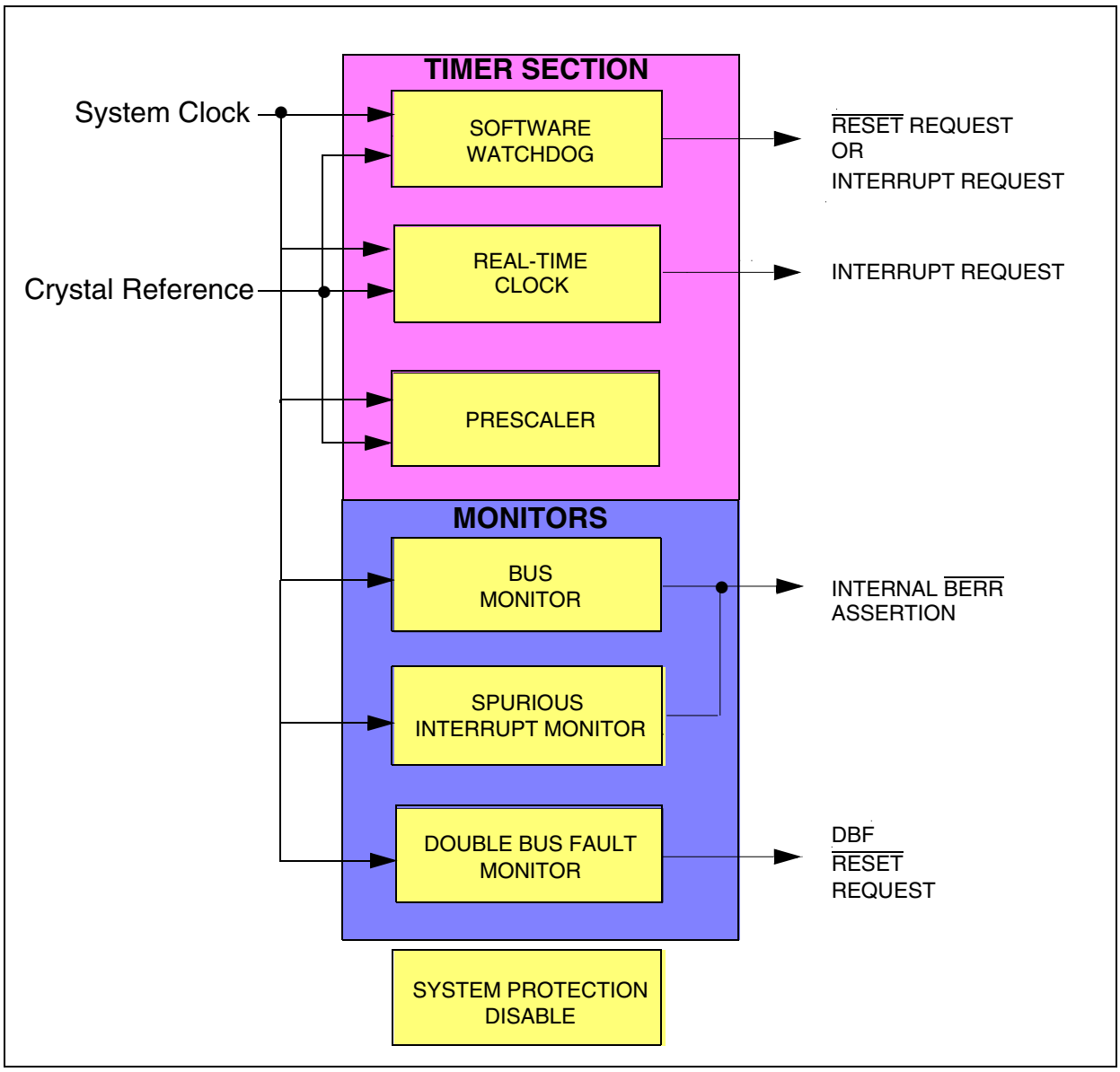


Figure 3-13 System Protection Block Diagram

3.5.1 Timer Section Functions

This section is an explanation of the features represented in **Figure 3-13**. The timer section includes the software watchdog, the real-time clock and the prescaler.

3.5.1.1 Software Watchdog Interval Timer (SWDOG)

The software watchdog monitors system software by requiring the software to periodically reload the software watchdog interval timer to prevent a time-out alarm from being issued and to provide a controlled exit from runaway software loops. The software watchdog is prevented from issuing a time-out alarm by requiring application

software to execute a special programming sequence to reload the SWDOG interval timer on a regular basis to inhibit time-out. If this real-time service does not take place, the software watchdog will issue either a reset or an interrupt request.

Features of the software watchdog include:

- Ability to select between an interrupt request or a reset if a time-out occurs.
- The input clock to the software watchdog interval timer may be the system clock, the crystal frequency, or one of four prescaler taps.
- User-specified time-out intervals which range from one µs to 67 seconds (with a 2-MHz crystal reference frequency input) or from 60.6 ns to 4.1 seconds (with a 33-MHz system clock input). See **Table 3-76** for a complete listing of time-out interval ranges for the software watchdog (SWDOG) at selected frequencies.
- The software watchdog interval timer is readable at any time.
- The software watchdog and the real-time interval timers can be configured to create a 32-bit interval timer for extended time-out.

3.5.1.2 Real-Time Clock (RTC)

The real-time clock provides a periodic interrupt timer (PIT). The features of the real-time clock include:

- Periodic interrupt capability if programmed to generate an interrupt request at time-out.
- The input clock to the real-time interval timer may be the system clock, the crystal frequency, or one of four prescaler taps.
- User-specified time-out intervals which range from 1 µs to 67 seconds (with a 2-MHz crystal reference frequency input) or from 60.6 ns to 4.06 seconds (with a 33-MHz system clock input). See **Table 3-76** for a complete listing of time-out interval ranges for the real-time clock at selected frequencies.
- The real-time clock interval timer is readable at any time.
- The RTC timer is user programmable to load a new interval value in one of two ways: load a new time interval immediately after a write to the interval register or load the new interval value after current interval completes.

3.5.2 SWDOG and RTC Configured as 32-Bit Interval Timer

The real-time clock and the SWDOG may be chained together to form a 32-bit interval timer. The most significant word of the timer is the software watchdog, and the least significant word of the timer is the real-time clock interval timer. In this configuration, the SWDOG interval timer acts as a 16-bit extension to the real-time clock interval timer and normal software watchdog functionality is disabled. See **Figure 3-19**.

The 32-bit interval timer features include:

- Exceptionally long time-out intervals by chaining the two 16-bit interval timers.
- The interval timers are readable at any time.
- The interval timers are unaffected by master reset so that timer configuration is insensitive to resets while the MCU is operating.
- A selectable clock source: system clock, crystal reference frequency or one of

four prescaler taps.
- In chained mode, the 32-bit timer can only assert an interrupt request as a result of time-out.
- The 32-bit interval timer can be user programmed in one of two ways: load a new time interval immediately after write to interval register or load register after current interval completes.

3.5.2.1 Prescaler

The SWDOG and the RTC both share a 10-bit synchronous timer prescaler.

The system protection prescaler features include:

- Clock source which is selectable between the PLL reference crystal frequency and the system clock.
- Four prescaler taps which add to the clock choices that can be applied to the SWDOG or the RTC interval timers.
- The prescaler may be read at any time.
- The prescaler is reset to zero under software control so that a complete interval time-out value will occur when the SWDOG and/or RTC interval timers begin counting.

3.5.3 Monitor Functions

This section documents the functionality of the bus monitor, the double bus fault monitor, and the spurious interrupt monitor.

3.5.3.1 Bus Cycle Timeout Monitor

If a bus cycle does not terminate with the assertion of DTACK or BTACK, the bus monitor will assert an internal BERR to terminate the bus cycle. The bus monitor may monitor internal and/or external bus accesses. Four selectable time-out intervals range from eight system clock cycles to 64 system clock cycles.

Typical asynchronous bus systems require the implementation of some kind of watchdog to monitor excessive DTACK or BTACK signal response times during a bus access. If an unimplemented address is accessed or a peripheral is inoperative, the DTACK or BTACK will not be issued and the bus will be locked up while it waits for the required response.

The BIM provides a bus monitor to monitor all internal bus accesses and an option to monitor any internal to external bus accesses. The bus monitor watches the DTACK response time during a normal bus cycle (or BTACK during a burst cycle) or watches for the auto vector (AVEC) signal during an interrupt acknowledge (IACK) bus cycle. If the response time of the internal modules or external peripherals exceeds the programmed count, the bus monitor will assert an internal bus error (BERR). The bus monitor does not watch for DTACK or BTACK response on the external bus unless the MCU initiates the bus cycle.

The bus monitor monitors the DTACK, BTACK or AVEC response time (in clock cycles) by using a progammable maximum allowable response interval.

Since the bus monitor is concerned only with internally initiated bus cycles, it will be disabled when the external bus request (EBR) pin is asserted.

There are four selectable response time intervals for the bus monitor ranging from eight system clock cycles (~142 ns @ 33 MHZ) to 64 system clock cycles (~1.94 µs @ 33 MHz). The intervals are selectable with the BMT field in the system protection control register (see **Table 3-67**). The programmability of the time-out allows for varying external peripheral response times. The timing mechanism is derived from taps off a six stage divider chain clocked by the system clock. The taps are taken at the “÷8”, “÷16”, “÷32”, and “÷64” stages. The divider chain is cleared and started on all internal cycles, and optionally on internal to external cycles, and is then clocked by the system clock. If the cycle is not terminated by DTACK or AVEC within the selected response time, or BTACK response is too slow, a time-out will occur. When a time-out occurs, the bus monitor asserts internal BERR to terminate the bus cycle.

The BME bit in the system protection control register (SYPCR), when set, will alert the internal bus monitor to watch DTACK and BTACK response time during an internal to external bus cycle. Refer to **Table 3-67**.

3.5.3.2 Double Bus Fault Monitor (DBF)

The double bus fault monitor asserts reset if a double bus fault occurs. The DBF monitor monitors the HALT line of the intermodule bus which some CPUs assert after a double bus fault occurs.

A double bus fault occurs if internal BERR is asserted by the monitor during the process of servicing particular exceptions which are specific to the CPU ***(see appropriate CPU Manual)***. The CPU is halted and all processes cease. The halted state is the double bus fault state. Entry into this state (CPU initiated HALT) triggers the double bus fault monitor which causes a master reset. A flag is set in the BIM module’s reset status register (RSR) when reset is caused by a double bus fault. The operation of this function may be inhibited by the DBE bit in the system protection control register (refer to **Table 3-67**).

3.5.3.3 Spurious Interrupt Monitor (SPM)

The internal BERR signal is asserted if no interrupt arbitration takes place during an interrupt acknowledge (IACK) bus cycle (spurious interrupt request). This feature is always operating and cannot be disabled.

The internal spurious interrupt monitor will assert internal BERR if no activity is detected on the internal bus lines IARB0-IARB1 during an IACK cycle. During an IACK cycle, one or more internal modules will recognize the interrupt request level generated by the CPU and will arbitrate for the bus. Arbitration is carried out using the two lines IARB0 and IARB1 on the internal bus if neither of those lines are asserted during an IACK cycle, a spurious interrupt request has been encountered.

This feature can never be inhibited; it is always enabled.

3.5.4 Disabling The System Protection Sub-Module

For power savings or development purposes, the system protection sub-module may have all or part of its subsections disabled as described in the following sections.

3.5.4.1 SYSPROT Disable Bit

The clocks to the entire system protection sub-module may be shut down at any time by writing a '1' to the SYSPROT bit in the module disable register (MDR).

3.5.4.2 Timer Section Disable

The timer section may be disabled at any time by asserting the internal FREEZE line while the BIM MCR FRZ[0] bit is a '1'. The clock to the software watchdog, the RTC and prescaler will be disabled on the low phase of the system clock. It will remain disabled until the next rising edge of the system clock after the FREEZE line is negated.

3.5.4.3 Bus Monitor Disable

When the internal FREEZE line is asserted and the BIM MCR FRZ[1] bit is a '1', the clock to the bus monitor will stop and therefore disable BERR from occurring if DTACK or BTACK is later than the programmed bus monitor time-out interval. The clock will be disabled on the low phase and will remain disabled until the next rising edge of the system clock after the FREEZE line is negated.

3.5.5 Programming Model

Table 3-66 shows the address map for this BIM submodule. These registers provide status information, and configuration and operation control.

Table 3-66 System Protection Submodule Programming Model

Address	15 8	7 0
0xYF FA50	SYSTEM PROTECTION CONTROL REGISTER (SYPCR)	
0xYF FA52	TIMER CONTROL REGISTER (TIC)	TIMER INTERRUPT VECTOR REGISTER (TIV)
0xYF FA54	RESERVED	SOFTWARE WATCHDOG SERVICE REGISTER (SWS)
0xYF FA56	PRESCALER (PRE) [READ-ONLY]	
0xYF FA58	SOFTWARE WATCHDOG INTERVAL REGISTGER (SWI)	
0xYF FA5A	REAL-TIME CLOCK INTERVAL REGISTER (RTI)	
0xYF FA5C	SOFTWARE WATCHDOG INTERVAL TIMER (SWIT) (READ-ONLY)	
0xYF FA5E	REAL-TIME INTERVAL COUNTER (RTIT) (READ-ONLY)	

3.5.6 System Protection Registers

The system protection registers consist of:

- System protection control register (SYPCR),
- Timer control register (TIC),
- Timer interrupt vector register (TIV),
- Software watchdog service register (SWS).

3.5.6.1 System Protection Control Register (SYPCR) (V4 and V5)

SYPCR — System Protection Control Register **0xYF FA50**

MSB 15	14	13	12	11	10	9	LSB 8
REN	SREN	PCLK	SLPC	RZ	SZ	IRSEL	TIEN
RESET:							
0	0	0	0	0	0	0	0
MSB 7	6	5	4	3	2	1	0
LCI[1]	TIQL[2]	TIQL[1]	TIQL[0]	DBE	BME	BMT[1]	BMT[0]
RESET:							
1	0	0	0	0	0	0	0

NOTES:
1. The LCI bit is enabled only on V5.

Bits [15:7] of the system protection control register are reset or asserted by power-on reset and bits [6:0] are reset or asserted by master reset.

NOTE

When FREEZE is asserted, bits [3:0] are writable at any time. Otherwise, these bits may be written only once after a power-on reset or master reset.

Table 3-67 SYPCR Bit Descriptions

Bit(s)	Name	Description
15	REN	RESET enable. This bit controls whether writes to the real-time interval register will reset the prescaler (see **3.5.9.1 Real-Time Interval Register (RTI)**). This control bit allows the user to control whether the Prescaler is started over at zero upon writing a new value to the real-time interval (RTI) register. Master reset will not affect this bit. 0 = Prescaler reset disabled 1 = Prescaler reset enabled
14	SREN	SWDOG reset enable. This bit controls whether writes to the Software Watchdog Interval (SWI) register will reset the prescaler. See **3.5.6.4 SWDOG Service Register (SWS)**. This control bit allows the user to control whether the prescaler is started over at zero upon writing a new value to the software watchdog interval (SWI) register. Master reset does not affect this bit. 0 = Prescaler reset disabled 1 = Prescaler reset enabled

Table 3-67 SYPCR Bit Descriptions (Continued)

Bit(s)	Name	Description
13	PCLK	Prescaler clock select. The PCLK bit selects the clock source to the prescaler. Master reset will not affect this bit. A clock source may be selected whether or not that source is active, (i.e., the PLL or crystal is turned off). In this case, the system protection prescaler will not run since it will have selected a channel which is not carrying a clock signal. If the SWDOG and the RTC interval timers are using the system protection prescaler taps, or the crystal frequency, they also will not run. **WARNING**: Because of the asynchronous relationship between the system clock and crystal reference clock inputs to the prescaler, switching these clock inputs while the prescaler is running (SWIT or RTIT being clocked from a prescaler tap) may cause erroneous counts. 0 = Selects system clock divided by 2 1 = Selects crystal reference frequency divided by 2
12	SLPC	Software watchdog LPSTOP count bit. This bit controls whether the software watchdog stops counting after entering LPSTOP mode. This bit is unaffected by master reset. 0 = SWDOG does not count in LPSTOP; counting will resume when exit from LPSTOP 1 = SWDOG counts in LPSTOP
11:10	RZ, SZ	Timer zero flags. The timer zero flags function as the interrupt pending bits for the interval timers. By reading the flag as a '1' and then writing the flag to a zero, the flag is cleared. When the flag is cleared, the pending interrupt request is cleared. Should another time-out event occur after the user has read the flag as a '1' and before the user writes a '0' to it, the flag will remain a '1' regardless of the attempt to clear it. This feature provides limited double buffering of time-out event. These bits are unaffected by master reset. RZ is asserted after the real-time interval timer reaches zero; SZ is asserted after the software watchdog interval timer reaches zero.
9	IRSEL	Interrupt/reset select bit. The IRSEL bit selects whether the software watchdog interval timer (SWIT) issues an interrupt request or a master reset when the SWDOG interval timer times out. This bit is unaffected by master reset. 0 = SWIT issues reset 1 = SWIT issues the interrupt request level programmed in the TIQL field of the system protection control register
8	TIEN	Timer interrupt enable. The TIEN bit controls whether the real-time interval timer (RTIT) issues an interrupt request after timing-out. The interrupt request priority is specified in bits [6:4] of the system protection control register. This bit is unaffected by master reset. 0 = No interrupt request is issued upon RTIT reaching zero 1 = Interrupt request will be issued upon RTIT reaching zero
7	LCI	Load count immediate. The LCI bit controls when a change in real-time interval register (or real-time interval register and software watchdog interval register in 32-bit chained mode) is loaded into the real-time interval timer (or real-time interval timer and SWDOG interval timer in 32-bit chained mode). This bit is unaffected by master reset. The LCI bit is enabled only on V5. 0 = Count is allowed to reach zero (time-out) before the new value is loaded 1 = new value is loaded immediately after the writing a new value to the RTI register
6:4	TIQL	Timer interrupt request level field. Bits [6:4] of the system protection control register define the priority of interrupt request that is generated when the real-time interval timer or the software watchdog interval timer time-out. A value of '000' disables interrupt requests, but does not disable the interval timers from running. After master reset, the timer interrupt request level field defaults to *interrupt disabled*. 000 = Disable interrupt requests 001 = Interrupt request level 1 010 = Interrupt request level 2 011 = Interrupt request level 3 100 = Interrupt request level 4 101 = Interrupt request level 5 110 = Interrupt request level 6 111 = Interrupt request level 7

Table 3-67 SYPCR Bit Descriptions (Continued)

Bit(s)	Name	Description
3	DBE	Double bus fault monitor enable. This bit controls the monitoring for double bus fault on the inter-module bus (IMB) by the double bus fault monitor (see **3.5.3.2 Double Bus Fault Monitor (DBF)** for a complete explanation of a double bus fault). When test mode is enabled or FREEZE is asserted, this bit is writable at any time. Otherwise, this bit may be written only once after a power-on reset or master reset, after which no more writes will be allowed. 0 = Disable double bus fault monitor 1 = Enable double bus fault monitor
2	BME	Bus monitor external enable. This bit controls whether the internal bus monitor will operate during an external bus cycle. When enabled, the bus monitor monitors DTACK or BTACK response time for all bus cycles. When test mode is enabled or FREEZE is asserted, this bit is writable at any time. Otherwise, this bit may be written only once after a power-on reset or master reset. 0 = Disable bus monitor function for external bus cycle 1 = Enable bus monitor function for external bus cycle
1:0	BMT	Bus monitor timing. These bits select the time-out interval, in system clocks, for the bus monitor. After master reset, the bus monitor time-out value defaults to *64 system clocks*. For external accesses, this time-out value should be longer than the chip select access time. When FREEZE is asserted, these bits are writable at any time. Otherwise, these bits may be written only once after a power-on reset or master reset. 00 = 64 system clocks 01 = 32 system clocks 10 = 16 system clocks 11 = 8 system clocks

3.5.6.2 Timer Control Register (TIC)

TIC — Timer Control Register **0xYF FA52**

MSB 7	6	5	4	3	2	1	0
PRR	SWC[2]	SWC[1]	SWC[0]	D2R	RTC[2]	RTC[1]	RTC[0]
RESET:							
0	0	0	0	0	0	0	0

The timer control register contains the prescaler reset bit (PRR), the real-time clock control field (RTC[2:0]), the divide-by-2 reset bit (D2R), and the software watchdog clock control field (SWC[2:0]). The SWC and RTC fields select the clock input to the software watchdog interval timer and the real-time interval timer, respectively. Master reset will have no effect on this register. **Figure 5-2** shows the prescaler block diagram. When the user selects the 32-bit chained mode, normal software watchdog functionality is disabled. For more details, see **3.5.9.9 RTC and SWDOG in 32-Bit Mode**.

The system clock and the crystal reference settings allow clocking of the SWDOG or the RTC interval timers by the system clock divided by two, or the crystal reference divided by two, respectively. The prescaler settings, on the other hand, allow one of four prescaler taps to clock the interval timers. When neither the RTC nor SWDOG select a prescaler tap, the prescaler is disabled from counting.

Asserting reset to exit LPSTOP has no affect on either the SWC or the RTC fields. Thus, the SWDOG and the RTC can each continue to run with the same clock configuration in and out of LPSTOP. These bits are unaffected by master reset.

Table 3-68 TIC Bit Descriptions

Bit(s)	Name	Description
7	PRR	Prescaler reset. This bit resets the prescaler when it is written to a '1'. This capability is provided so that the prescaler may be reset at the moment the user is changing the configuration of the timer control register. The user may not wish to reset the prescaler in cases where the SWDOG and the RTC are both using prescaler taps as clock inputs. A prescaler reset for the sake of one interval timer would affect the count in the other. In this case, a '1' should NOT be written to the PRR bit. The PRR bit always reads '0'.
6:4	SWC[2:0]	SWDOG clock control. Provides seven possible settings for choosing the input clock for the software watchdog interval timer (SWIT). 000 = System clock/2 001 = Crystal reference/2 010 = 2^{10} prescaler tap 011 = 2^{8} prescaler tap 100 = 2^{6} prescaler tap 101 = 2^{4} prescaler tap 110 = RTC extension 111 = SWDOG off
3	D2R	Divide-by-2 RESET. This bit resets the input clock divide-by-2 timer when it is written to a '1'. This capability is provided so that the divide-by-2 timer may be reset at the moment the user is changing the configuration of the timer control register. The user may not wish to reset the divide-by-2 timer in cases where the SWDOG and the RTC are both in use. A divide-by-2 timer reset for the sake of one interval timer would affect the count in the other. In this case, a '1' should NOT be written to the D2R bit. The D2R bit always reads '0'.
2:0	RTC{2:0]	RTC clock control. Provides six possible settings for choosing the input clock for the real-time interval timer (RTIT). 000 = System clock/2 001 = Crystal reference/2 010 = 2^{10} prescaler tap 011 = 2^{8} prescaler tap 100 = 2^{6} prescaler tap 101 = 2^{4} prescaler tap 110 = RTC off 111 = RTC off

3.5.6.3 Timer Interrupt Vector Register (TIV)

TIV — Timer Interrupt Vector Register **0xYF FA53**

MSB 7	6	5	4	3	2	1	0
TIV[7]	TIV[6]	TIV[5]	TIV[4]	TIV[3]	TIV[2]	TIV[1]	TIV[0]
RESET:							
0	0	0	0	1	1	1	1

Table 3-69 TIV Bit Descriptions

Bit(s)	Name	Description
7:0	TIV[7:0]	Timer interrupt vector. The timer interrupt vector register (8 bits) contains the vector number that is fetched during an interrupt acknowledge (IACK) cycle in response to a SWDOG or RTC interrupt request. This register is readable and writable at any time. Master reset will have no effect on this register.

3.5.6.4 SWDOG Service Register (SWS)

SWS — SWDOG Service Register **0xYF FA55**

MSB 7	6	5	4	3	2	1	0
SWS[7]	SWS[6]	SWS[5]	SWS[4]	SWS[3]	SWS[2]	SWS[1]	SWS[0]
RESET:							
0	0	0	0	0	0	0	0

The software service register is the location to which the SWDOG servicing sequence is written. See **3.5.9.2 Software Watchdog Interval Timer Operation (SWIT)** for instructions on the use of this register. This register is unaffected by master or power-on reset Reads of this register will always return a zero.

Table 3-70 SWS Bit Descriptions

Bit(s)	Name	Description
7:0	SWS[7:0]	SWDOG service. The software service register is the location to which the SWDOG servicing sequence is written. See **3.5.9.2 Software Watchdog Interval Timer Operation (SWIT)** for instructions on the use of this register. This register is unaffected by master or power-on reset Reads of this register will always return a zero.

3.5.7 Timer Section

The timer section consists of the prescaler (PRE), the SWDOG and RTC clock muxes (SMUX and RMUX), the SWDOG and RTC interval timers (SWIT and RTIT), and the SWDOG and RTC interval registers (SWI and RTI).

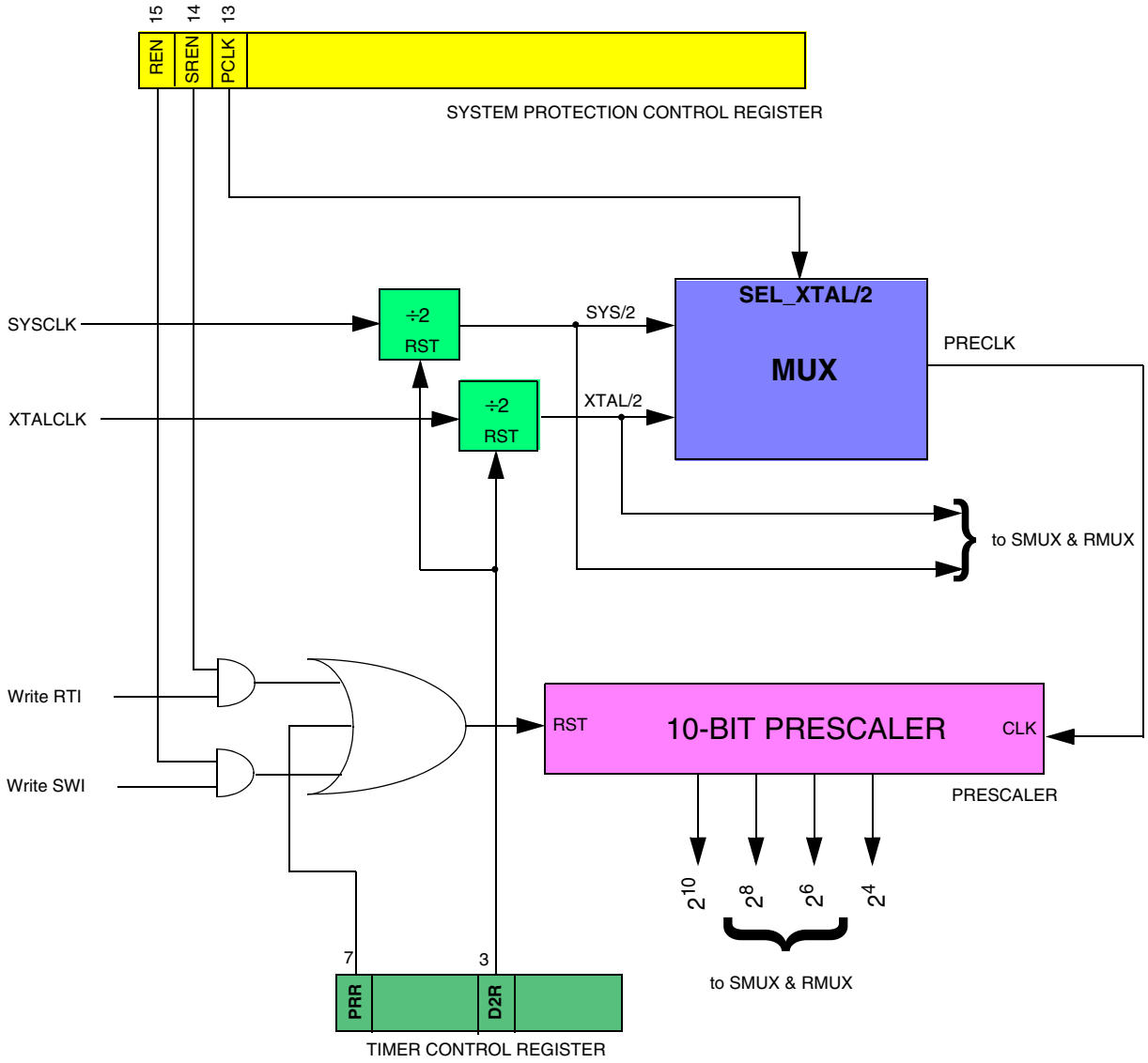


Figure 3-14 Software Watchdog, Real-Time Clock and Prescaler

3.5.7.1 System Protection Prescaler (PRE)

PRE — System Protection Prescaler Register **0xYF FA56**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-71 PRE Bit Descriptions

Bit(s)	Name	Description
15:0	0	SWDOG service. The SWDOG and the RTC both share a 10-bit synchronous timer which is known as the prescaler. Four different prescaler taps can be used as clocks to the SWDOG or RTC interval timers. Regardless of which clock is selected to feed the prescaler (see **3.5.2.1 Prescaler**), it is reiterated here that the clock will have been divided by two before reaching the prescaler (see **Figure 3-14**). The prescaler register is readable at any time. Since the prescaler is 10 bits, the unused bits [15:10] in the prescaler register will always be zero. If either of the reset enable bits (REN or SREN) are set, writes to the corresponding interval register will reset the prescaler. The prescaler is also reset whenever the PRR bit in the timer control register (TIC) is written to a '1'. The prescaler is unaffected by master reset and cannot be written.

3.5.7.2 Timer Section Disable

The clocks to the entire system protection sub-module may be disabled at any time by writing a '1' to the SYSPROT bit in the module disable register. See **3.1.11 Module Disable Register (MDR)** for more information on this control bit.

The timer section alone may be disabled at any time by asserting the FREEZE line while the BIM MCR FRZ[0] bit is a '1'. The clock to the software watchdog, the RTC and the prescaler will be disabled on the low phase of the system clock. It will remain disabled until the next rising edge of the system clock after the FREEZE line is negated.

3.5.8 Software Watchdog and Real-Time Clock Interrupts

The timer section has the highest priority of all the interrupt sources in the BIM. If the BIM wins the interrupt acknowledge cycle and the SWDOG or RTC has a pending interrupt request at that level, the timer interrupt vector (TIV) is placed on the intermodule bus (IMB).

3.5.9 SWDOG Interval Register (SWI)

SWI — SWDOG Interval Register **0xYF FA58**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-72 SWI Bit Descriptions

Bit(s)	Name	Description
15:0	0	Software watchdog interval. The software watchdog interval register (SWI) contains the value to be loaded into the software watchdog interval timer (SWIT) immediately upon the lower byte of the SWI being written. If the SWDOG is configured to interrupt at time-out (IRSEL = 1), the SWI value will be loaded into the SWIT every time it counts down to zero. A write to either the lower byte or the 16-bit word of the SWI activates the SWIT. This register is readable and writable at any time. Writes to this register will reset the prescaler if the SREN bit in the system protection control register (SYPCR) is set. A zero written to this register will turn off and clear the software watchdog interval timer without issuing a reset or interrupt request. The SWI is unaffected by master reset. This register will therefore contain the value programmed into it prior to master reset so that counting may continue during reset or freeze operation. The SWP is unaffected by master reset. This register will therefore contain the value programmed into it prior to master reset. Power-on reset will reset this register to zero.

3.5.9.1 Real-Time Interval Register (RTI)

RTI — Real-Time Interval Register **0xYF FA5A**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-73 RTI Bit Descriptions

Bit(s)	Name	Description
15:0	0	Real-time interval. The real-time interval register (RTI) contains the value to be loaded into the real-time interval timer (RTIT) when the lower byte of the RTI is written, if the LCI bit is set, and every time the RTIT counts down to zero. A byte write to the lower byte of this register or a 16-bit write of the RTI activates both the SWDOG and RTC timers in 32-bit interval timer mode or the RTIT when not in 32-bit interval timer mode. This register is readable and writable at any time. Writes to this register will reset the prescaler if the REN bit in the system protection control register (SYPCR) is set. A zero written to this register will turn off and clear the real-time interval timer without issuing an interrupt request. (If LCI = 0, the timer will not stop until next time-out). The RTI is unaffected by master reset. This register will therefore contain the value programmed into it prior to master reset so that counting may continue during reset or freeze operation.

3.5.9.2 Software Watchdog Interval Timer Operation (SWIT)

SWIT — Software Watchdog Interval Timer Operation Register **0xYF FA5C**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-74 SWIT Bit Descriptions

Bit(s)	Name	Description
15:0	0	Software watchdog interval timer operation. The SWDOG consists of a 16-bit interval timer clocked by the output of the SWDOG clock mux (SMUX). The SMUX output is applied to the SWIT which will start counting down when loaded with a non-zero value from the SWI register. The SWIT cannot be written directly. When accessing the SWI register with byte operations, the value in the SWI register will not be loaded into the SWIT until the lower bits [7:0] of the SWI are written. When the software watchdog times out, depending on the state of the IRSEL bit in the system protection control register, it will assert a reset or an interrupt request. If it is programmed to generate an interrupt request, the SWDOG interval timer will then load the current value from the SWI register and begin counting down again. The user may change the time out value in the software watchdog interval register at any time. After time-out occurs, the software watchdog zero flag (SZ) will be asserted. The software watchdog interval timer is readable at any time and writes have no effect. When master reset is asserted, the software watchdog interval timer is unaffected. See **Figure 3-15**.

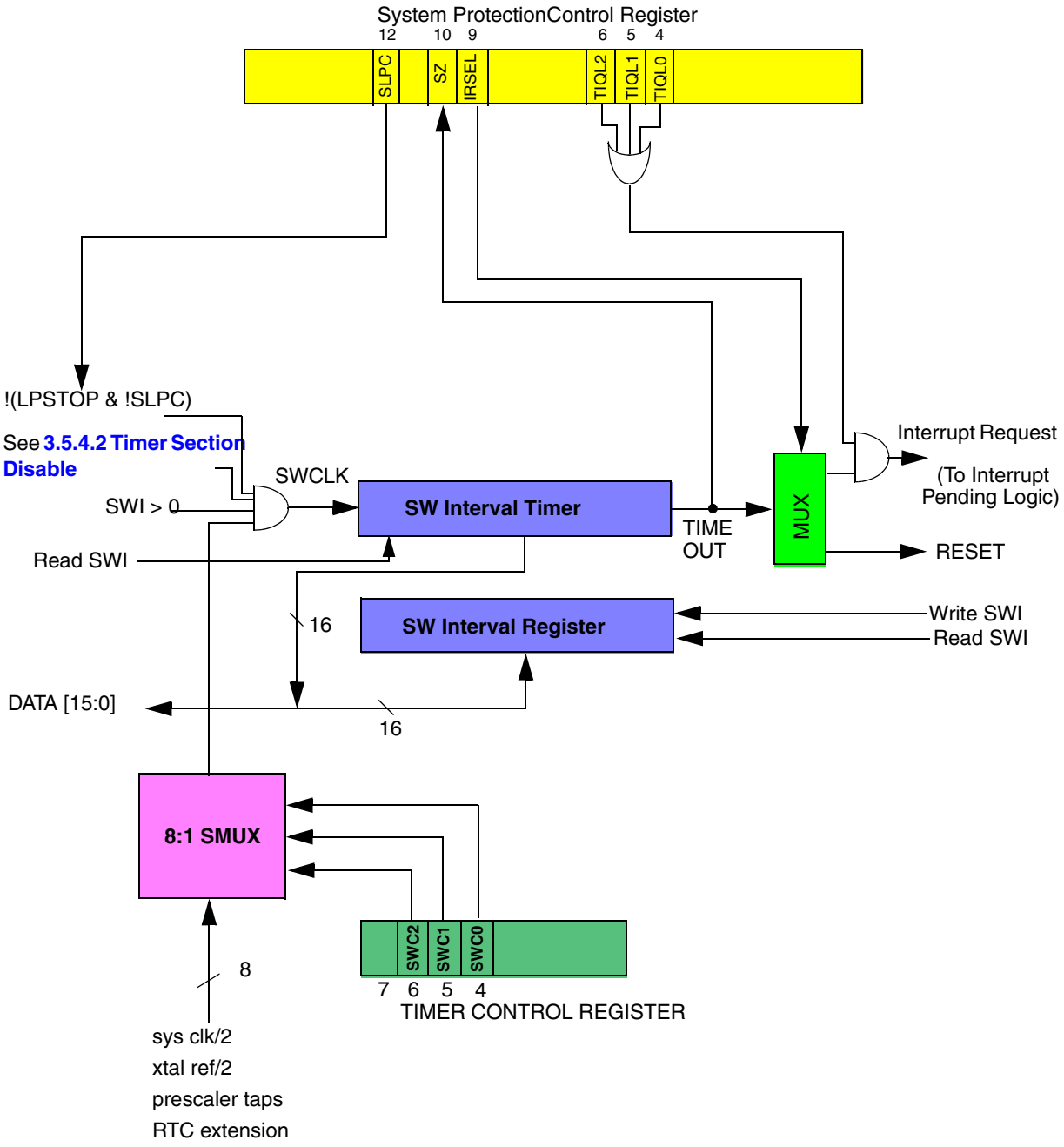


Figure 3-15 Software Watchdog Timer Block Diagram

3.5.9.3 Software Watchdog Time-Out Interval

Although most software disciplines encourage or permit the watchdog concept, all systems do not need the same time-out intervals.

The software watchdog time-out interval is set by the software watchdog interval register (SWI) and the speed of the clock applied to the SWDOG interval timer (SWIT). The clock applied to the SWIT, SWCLK, is derived from one of the following

- A clock operating at the crystal reference frequency divided by two.
- A clock operating at the system clock frequency divided by two.
- A prescaler tap — PRECLK10, PRECLK8, PRECLK6, or PRECLK4, which is system clock divided by two or the crystal frequency divided by two, and again divided by one of the prescaler tap values.

The selection of the clocks just mentioned, used to derive SWCLK, is controlled by the SWC field in the timer control register and by the PCLK bit in the system protection control register.

3.5.9.4 SWDOG Interval Calculation

The software watchdog provides a wide range of programmable time-out intervals depending on the system clock or crystal reference frequency and the input frequency chosen (refer to **Table 3-76**).

The SWDOG interval (not prescaled) is calculated as follows:

SWDOG interval = (SWI value) * 2 / (CLOCK)
where CLOCK = system clock or crystal reference frequency (whichever is chosen in the SWC field of the timer control register).

EXAMPLE

CLOCK = 2MHz (crystal ref freq) and SWI value = 256
SWDOG interval = (256 * 2) / 2 MHz = 256 uS

The SWDOG Interval (prescaled) is calculated as follows:

SWDOG interval = (SWI value) * 2 / (CLOCK / prescaler)
where CLOCK = system clock or crystal reference frequency to the prescaler (whichever is chosen in the PCLK bit of the system protection control register).

and prescaler = prescaler tap setting in the SWC field of the timer control register.

EXAMPLE

CLOCK = 33 MHz (system clock freq), SWI value = 1024,
and prescaler = 64
SWDOG interval = (1024 * 2) / (33 MHz / 64) = 4 ms

which gives the ranges in **Table 3-76**.

3.5.9.5 SWDOG Service Sequence

The special service sequence which inhibits the software watchdog time-out, and reloads the software watchdog interval timer with the value in the software watchdog interval register, consists of two steps:

- Write 0x55 to the SWDOG service register (SWS)
- Write 0xAA to the SWDOG service register (SWS)

Both writes must occur in the proper order and prior to time-out. Any number of instructions, up to the end of the time-out interval, may be executed in between each write.

When the proper service sequence occurs, the software watchdog interval timer reloads the value of the SWDOG Interval register and the process begins again.

If the time-out request is reached before a service action, the SWDOG will cause a master reset (or an interrupt request, depending upon the state of the IRSEL bit).

3.5.9.6 Real-Time Interval Counter and RTC Operation (RTIT)

RTIT — Real-Time Interval Counter and RTC Operation Register **0xYF FA5F**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-75 RTIT Bit Descriptions

Bit(s)	Name	Description
15:0	0	Real-time interval counter and RTC operation. The real-time clock consists of a 16-bit interval timer clocked by the RTCLK output of the RTC clock mux (RMUX). The RTCLK signal is applied to the RTIT which will start counting down when a non-zero value is loaded from the RTI. The RTIT cannot be written directly. If the LCI bit is set, the RTI will not be loaded into the RTIT until the lower byte (or 16-bit word) of the RTI is written. If the LCI bit is 0, the timer will be allowed to time out before the RTI value is loaded. When the interval timer reaches zero, an interrupt request will be generated if the TIEN bit is set and if the Interrupt Request Level, TIQL, is nonzero. The real-time zero flag always sets, regardless of the TIEN or TIQL value. At this time, the real-time interval timer loads the current real-time interval register value and begins timing again. The real-time interval timer is readable at any time and writes have no effect. When master reset is asserted, the real-time interval timer is unaffected. See **Figure 3-16** for more information.

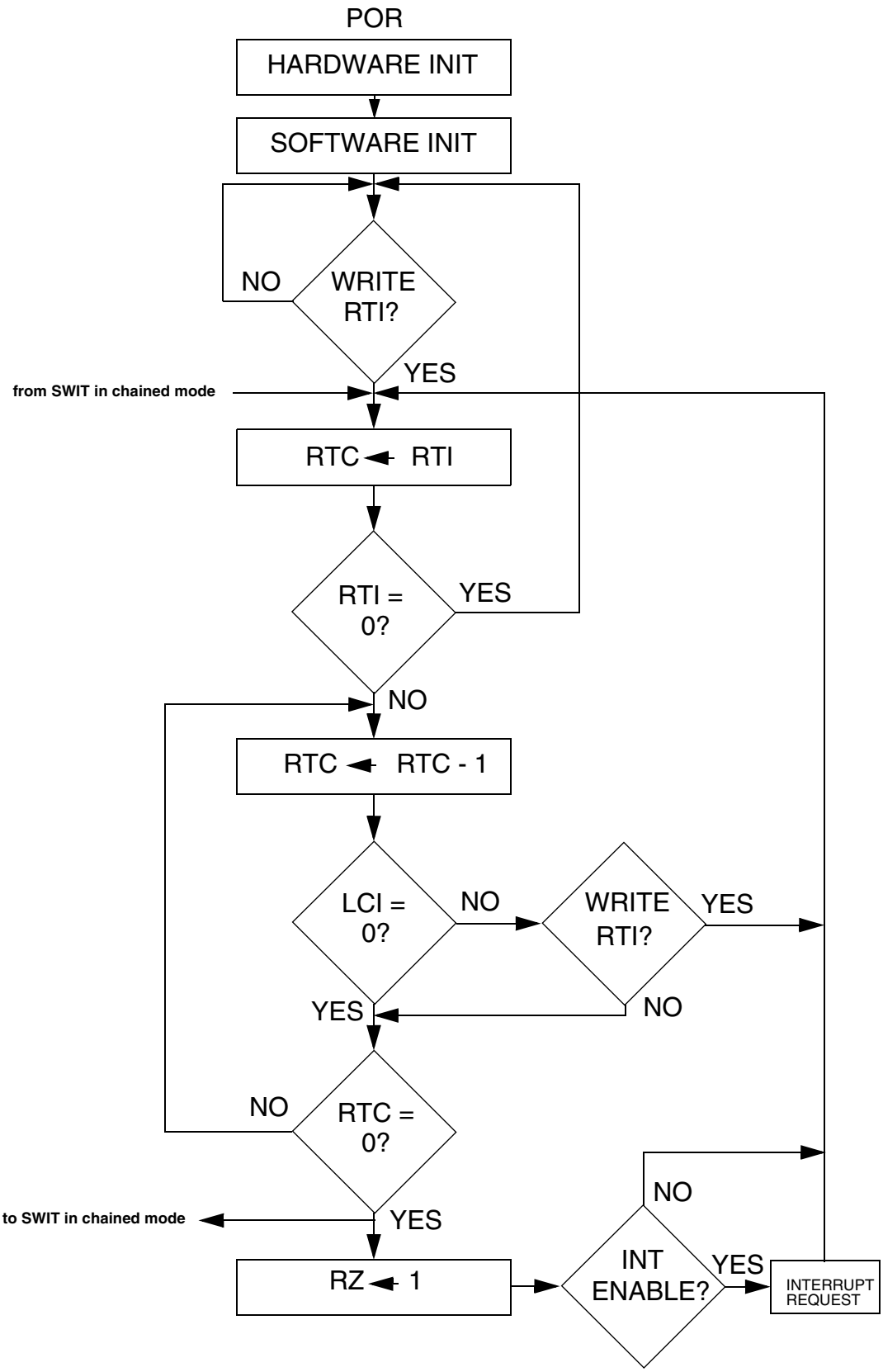


Figure 3-16 Real-Time Clock Flow

3.5.9.7 RTC Time-Out Interval

The real-time clock time-out is set by the real-time interval register (RTI) and the speed of the clock applied to the real-time interval timer (RTIT). The clock applied to the RTIT, RTCLK, is derived from one of the following

- A clock operating at the crystal reference frequency divided by two
- A clock operating at the system clock frequency divided by two
- A prescaler tap — PRECLK10, PRECLK8, PRECLK6, or PRECLK4, which is system clock divided by two or the crystal reference frequency divided by two, and again divided by one of the prescaler tap values.

The selection of the clocks is controlled by the RTC field in the timer control register and by the PCLK bit in the system protection control register.

3.5.9.8 RTC Interval Calculation

The real-time clock provides a wide range of programmable time-out intervals depending on the system clock or crystal reference frequency and the input frequency chosen (refer to **Table 3-76**).

The RTC interval (not prescaled) is calculated as follows:

RTC interval = (RTI value)x(2) / (CLOCK)
where CLOCK = system clock or crystal reference frequency (whichever is chosen in the RTC field of the timer control register).

The RTC interval (prescaled) is calculated as follows:

RTC interval = (RTI value)x(2) / (CLOCK / prescaler)
where CLOCK = system clock or crystal reference frequency to the prescaler (whichever is chosen in the PCLK bit of the system protection control register).

and prescaler = prescaler tap setting in the RTC field of the timer control register.

which gives the ranges in **Table 3-76**.

NOTE

The resolution of each range within a frequency category is its lower bound. For example, in the first row of the 2-MHz column, the resolution is one us.

A block diagram of the real-time clock is shown in **Figure 3-17**.

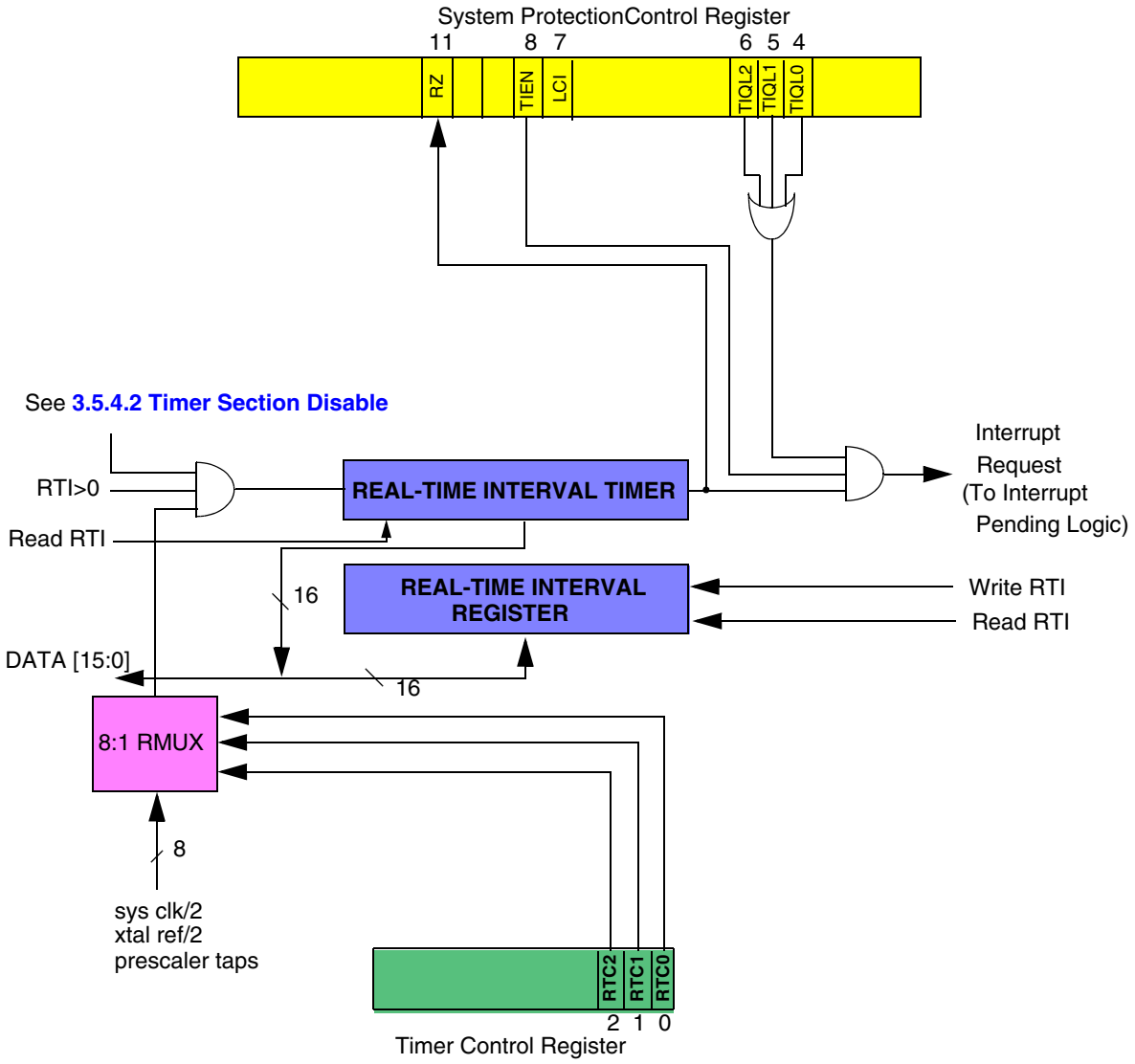


Figure 3-17 Real-Time Clock Block Diagram

Table 3-76 Examples of Timer Interval Ranges (Timers not Chained)

Prescaler Tap	Time-Out Ranges in Clock Frequency Categories			
	2 MHz	5 MHz	25 MHz	33 MHz
NONE	1 µs to 64.4 ms	400 ns to 26.2 ms	80 ns to 5.2 ms	60.6 ns to 3.96 ms
2^4	16 µs to 1 s	6.4 µs to 419.2 ms	1.28 µs to 83.2 ms	970 ns to 63.4 ms
2^6	64 µs to 4 s	25.6 µs to 1.68 s	5.12 µs to 332.8 ms	3.9 µs to 253.4 ms
2^8	256 µs to 17 s	102.4 µs to 6.71 s	20.5 µs to 1.33 s	15.5 µs to 1 s
2^{10}	1 ms to 67 s	409.6 µs to 26.83 s	81.92 µs to 5.32 s	62 µs to 4 s

3.5.9.9 RTC and SWDOG in 32-Bit Mode

The RTC and the SWDOG may be chained together to form a 32-bit interval timer. In this mode, the SWIT will act as the most significant word (MSW), and the RTIT will act as the least significant word (LSW). Upon reaching zero, rather than reloading the value in the interval register, the LSW will

- Count the SWIT down by one
- Roll over to 0xFFFF, and then
- Continue counting

One long word (32-bit) or two word (16-bit) values may be written to the SWI and RTI registers to trigger the interval timer chain.

The chained interval timer will not begin counting until the lower byte of the RTI (or 16-bit word) is written if the LCI bit is set. If the LCI bit is 0, the timer will be allowed to time out before a new RTI value is loaded.

Each time the real-time interval timer (RTIT) counts down through zero, a signal is sent to the software watchdog interval timer (SWIT) to count it down by one. When the SWIT and the RTIT have both reached zero, the real-time timer zero flag (RZ) will be set indicating that the interval timer chain has timed out. Upon timing out, provided that the TIEN bit is set and the TIQL bits are nonzero, an interrupt request will be asserted. The RTI value and the SWI value will then be loaded into each register's respective interval timer, and the chained timer will begin counting down again. The user may clear RZ, if necessary, by the standard clear mechanism.

Figure 3-19 is a flow diagram for the 32-bit chained mode operation.

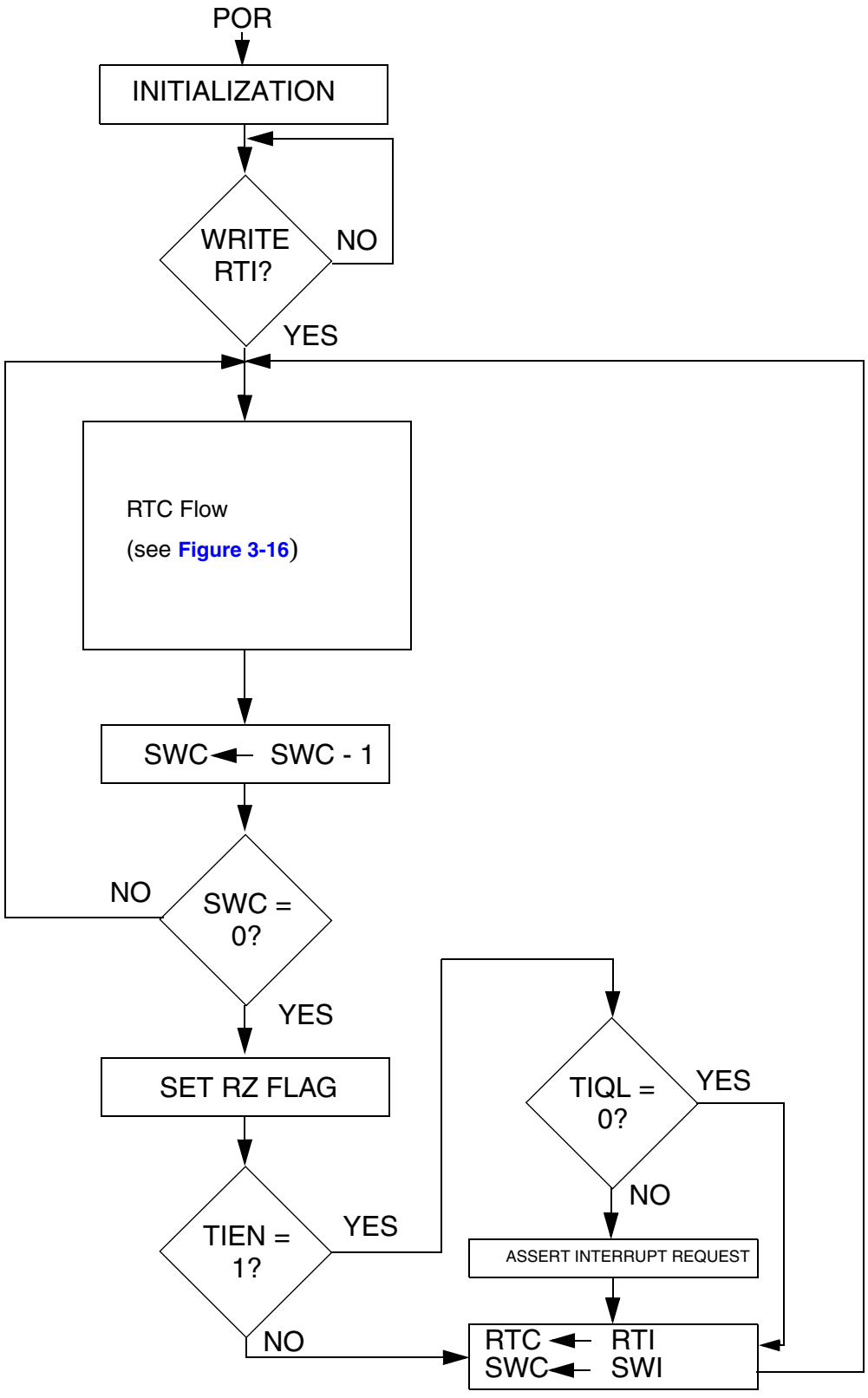


Figure 3-18 Flow Diagram of 32-Bit Chained Mode Operation

At any time during its operation, the timer chain may have a new interval value loaded into it by writing to the interval registers subject to the state of the LCI bit.

The user may read the timer at any time. However, separate, consecutive 16-bit reads may or may not be accurate, especially at times when the LSB is about to roll over and count the MSB down by one. In that particular case, consecutive 16-bit reads may result in as large as a 64-Kbyte error.

When master reset is asserted, the 32-bit timer's registers are unaffected.

3.5.9.10 32-Bit Timer Interval Calculation

The 32-bit timer provides time out intervals up to several days (refer to **Table 3-77**).

The 32-bit timer interval (not prescaled) is calculated as follows:

32-bit timer interval = (32 bit SWI/RTI value) x (2) / (CLOCK)
where CLOCK = system clock or crystal reference frequency (whichever is chosen in the RTC field of the timer control register).

The 32-bit timer interval (prescaled) is calculated as follows:

32-bit timer interval = (32 bit SWI/RTI) x (2) / (CLOCK / prescaler)
where CLOCK = system clock or crystal reference frequency to the prescaler (whichever is chosen in the PCLK bit of the system protection control register).

and prescaler = prescaler tap setting in the RTC field of the timer control register.

NOTE

The resolution of each range within a system frequency category is its lower bound. For example, in the first row of the 2-MHz column, the resolution is one us.

Table 3-77 Examples of Timer Interval Ranges (Timers Chained)

Prescaler Tap	Time-Out Ranges in System Clock Frequency Categories			
	2 MHz	5 MHz	25 MHz	33 MHz
NONE	1 µs to 1.2 hours	400 ns to 28.6 minutes	80 ns to 5.7 minutes	60.6 ns to 260.3 s
2^4	16 µs to 19 hours	6.4 µs to 7.64 hours	1.28 µs to 1.5 hours	969.6 µs to 1.2 hours
2^6	64 µs to 3.2 days	25.6 µs to 1.27 days	5.12 µs to 6.11 hours	3.9 µs to 4.6 hours
2^8	256 µs to 12.8 days	102.4 µs to 5.1 days	20.5 µs to 1 days	15.5 µs to 18.5 hours
2^{10}	1 ms to 51.2 days	409.6 µs to 20.36 days	89.92 µs to 4.1 days	62 µs to 3.1 days

3.5.9.11 Timer Application Example

This example shows the steps necessary to cause the timer to generate a level 5 interrupt request to exit LPSTOP after eight hours, in a system clocked by a 2-MHz crystal reference. A check of **Table 3-77** shows that at two MHz, we must use the 2^4 prescaler or higher. The timer control register (TIC) should be set to 0x101, thereby configuring

the RTIT to be clocked by the 2^4 prescaler and configuring the SWDOG timer to act as the RTC extension (see **3.5.6.2 Timer Control Register (TIC)**.

The SYPCR should then be set to 0xB15C in order to:

- Set the REN bit so that a write to the real-time interval register will reset the prescaler;
- Set PCLK = 1 to select the crystal frequency to clock the prescaler (see section **3.5.6.1 System Protection Control Register (SYPCR) (V4 and V5)**);
- Set the SLPC bit so that the SWDOG interval timer will continue to run in LPSTOP
- Set the TIEN bit to enable interrupt requests;
- Set an interrupt request priority level of five; and
- Leave the monitor settings in their default states (bits [3:0]).

NOTE

Writes to the IRSEL bit after selecting 32-bit timer mode will be ignored. Interrupt request generation is the default in 32-bit timer mode.

At this point, values must be calculated which will be loaded into the real-time interval register (RTI) and the SWDOG Interval register (SWI) to cause a time-out after eight hours.

Crystal frequency = 2 MHz

SWC & RTC interval	= 8 hours=	(SWI/RTI value*2)/(crystal freq/ 2^4)
	8 hours	= (SWI/RTI value*2) / (2e6 / 16)
SWI/RTI value		= 28800 s*(2e6 / 32)
		= 0x6B49D200 hex

The 32-bit interval timer will assert an interrupt request eight hours after the process began, and will count down to zero with resolution jumps of 16 us.

To make the interrupt request occur, the CPU interrupt mask must be dropped to a value below the value programmed into the TIQL bits (a 4 or lower). All that remains is to load the respective interval registers and the vector register and the timer will begin counting down as soon as the lower byte (or 16-bit word) of the RTI is written. When the RTI is loaded with the preset value, the prescaler will be reset and begin counting from precisely zero.

After eight hours, if the system was in LPSTOP mode, the 32-bit timer will issue an interrupt request which in turn will cause system wake-up from LPSTOP. The options at this point include using the interrupt handler to bring the SWDOG back on line, or writing the timer value to memory before reentering LPSTOP, or simply determining if the microcontroller's current state warrants a reset.

Figure 3-19 is a block diagram of the 32-bit mode operation.

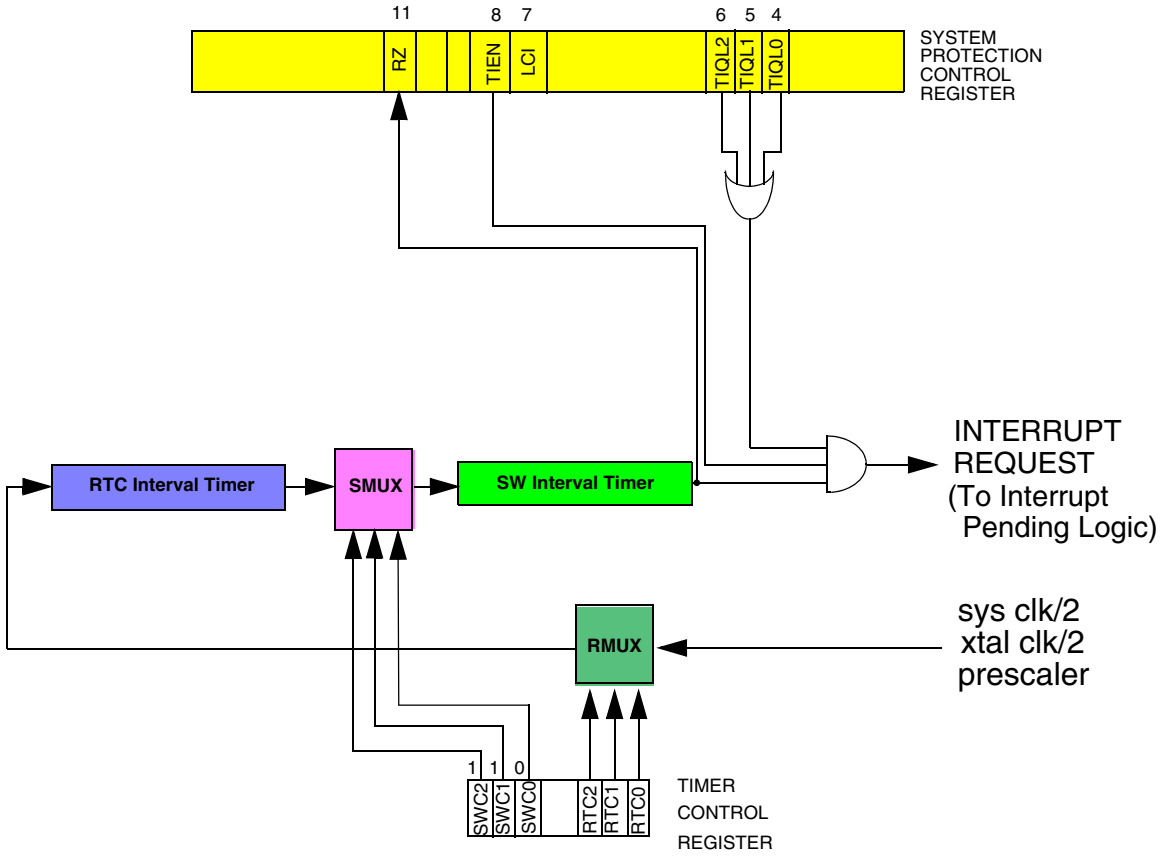


Figure 3-19 32-Bit Timer Operation Block Diagram

3.5.10 Low Power STOP Operation (LPSTOP)

When the burst integration module recognizes that the CPU has executed the “LPSTOP” instruction, the LPSTOP line is asserted.

NOTE

The LPSTOP line is *not asserted* when the CPU executes the “STOP” instruction and the RTC, SWDOG and monitors will continue to run.

3.5.10.1 RTC and LPSTOP Operation

The RTC always runs during LPSTOP unless disabled prior to executing the “LPSTOP” instruction. It will generate an interrupt request to bring the CPU out of LPSTOP if the interrupt request level (TIQL) is higher than the CPU mask level and if the TIEN bit is set. If the TIEN bit is not set, interrupt requests will not be asserted from the RTC. Refer to **3.5.6.1 System Protection Control Register (SYPCR) (V4 and V5)** for more information.

3.5.10.2 SWDOG and LPSTOP Operation

When LPSTOP is asserted, and the SLPC bit is not set, the software watchdog will be disabled on the low phase of the SWCLK, and will remain stopped until the LPSTOP line is negated. When LPSTOP negates, the timer resumes on the next rising edge of SWCLK. Otherwise, if the SLPC bit is set, the software watchdog will continue counting during LPSTOP.

The software watchdog may also be disabled during LPSTOP by writing a zero value to the SWI prior to executing the “LPSTOP” instruction.

If the system protection control register IRSEL bit is set, the software watchdog may generate an interrupt request to bring the CPU out of LPSTOP if the system protection control register interrupt request level (TIQL) is higher than the CPU mask level. If the IRSEL bit is low, however, the SWDOG may generate a reset to exit LPSTOP. Refer to **Table 3-67** for more information.

3.5.10.3 32-Bit Timer and LPSTOP Operation

Since the 32-bit timer is composed of the RTIT and SWIT, the SWIT must be configured to operate in LPSTOP, by setting the system protection control register (SYPCR) SLPC bit. Refer to **Table 3-67** for details.

As with the real-time clock, the 32-bit timer can be configured to interrupt out of LPSTOP if the system protection control register TIEN bit is set. See **Table 3-67** for more information.

3.5.10.4 Monitors and LPSTOP Operation

The bus monitor, double bus fault monitor and spurious interrupt monitor all respond to activity on the internal bus, and are effectively inactive as long as the system clock is disabled.

3.5.11 FREEZE Operation

When the FREEZE line is asserted, and the BIM MCR FRZ[0] bit is a ‘1’, the clock to the software watchdog, the real-time clock, and the prescaler will be disabled on the low phase of the system clock. The clock is only disabled within the system protection sub-module and its operation to the rest of the BIM is not affected. The clock will remain disabled until the FREEZE line is negated. It will be subsequently enabled to the system protection sub-module on the next rising edge of the system clock.

NOTE

The clock mentioned above can be either the system clock, or the crystal frequency.

When the FREEZE line is asserted and the BIM MCR FRZ[1] is a ‘1’, the bus monitor will stop counting and therefore disable BERR from occurring if DTACK or BTACK is later than the programmed bus monitor time-out interval.

3.5.12 RESET Determination

The system protection sub-module is reset when power-on reset (POR) is asserted.

Master reset will not reset bits [15:8] of the system protection control register, the timer control register, timer interrupt vector register, software watchdog interval register, real-time clock interval register, the prescaler, software watchdog interval timer, nor the real-time interval timer. The purpose here is to allow the configuration of these registers and bits to remain the same after reset so that the timer can continue to run in that configuration after a master reset.

The system protection sub-module provides two reset sources: the software watchdog and the double bus fault monitor. A reset from either source sets a flag in the BIM's reset status register (RSR).

3.6 Asynchronous Chip Select (ACS)

Typical systems require some external hardware to provide selects for external peripherals. Many of these devices do not have all the signals that are required for an asynchronous bus interface such as that of the M68000 family of parts. Additionally, higher levels of integration can provide cost, speed and reliability advantages over external logic. The asynchronous chip selects described here provide the means to minimize this external logic.

The primary function of the asynchronous chip select (ACS) submodule is to provide bus timing and bus cycle termination for external memory and peripheral devices. Each ACS can be programmed as read strobe, write strobe, output enable or interrupt acknowledge signals. The asynchronous chip selects may also be used to terminate external cycles, eliminating the external glue logic required to generate $\overline{\text{DTACK}}$.

There are seven ACS signals available. All ACS pins are optional and may not be included on a given MCU implementation incorporating the BIM. There is also a burst chip select channel which implements burst mode protocols for accessing burst memories.

3.6.1 Feature List

The following is a list of the features supported by the ACS module.

- Seven programmable asynchronous chip selects $\overline{\text{CS1}}$ — $\overline{\text{CS7}}$:
  - — Asynchronous chip selects may be independently programmed with the various selectable features.
- Various block sizes:
  - — The block size, which starts from the base address, can be programmed to be:
    64 Kbytes, 128 Kbytes, 256 Kbytes, 512 Kbytes, 1 Mbyte, 2 Mbytes, 4 Mbytes and 8 Mbytes.
- Support for 8-bit and 16-bit external devices:
  - — The port size can be programmed to be 8 or 16 bits. Eight-bit ports are accessible on both odd and even addresses when connected to D15-D8. Sixteen-bit ports can be accessed as odd-addressed bytes, even-addressed

bytes, or words. For byte transfers involving a 16-bit port, separate upper and lower byte asynchronous chip selects may be programmed.

- Read only, write only or read/write select:
  - A block of memory can be designated as read only, write only or read/write.
- Flexible asynchronous chip select assertion options
  - Asynchronous chip select signals assert in S0/S1($\overline{CE}$ timing) or the same bus state as $\overline{DS}$, so that control signals to memories or peripherals, such as $\overline{CE}$, $\overline{OE}$, and $\overline{WE}$, can be easily generated.
- Bus cycle termination with wait states:
  - This option allows DTACK to be generated internally to terminate the bus cycle. The desired number of wait states can be programmed by the user to interface with various devices.
- Support for slow bus interface device:
  - Additional timing flexibility allows asynchronous chip select signals to negate one clock before the DTACK generator ends the bus cycle.
- Address space checking:
  - Supervisor, user, program, data, and CPU space accesses can be optionally checked. The CPU space type is checked by programming the desired type in the proper base address register bits.
- IACK cycle support:
  - An asynchronous chip select can be programmed to match IACK cycles. The corresponding pin can be asserted for all IACK cycles or only for a specific interrupt priority level.

3.6.2 Asynchronous Chip Select Block Diagram

Figure 3-20 shows a programmable asynchronous chip select. All asynchronous chip selects have the same structure. All signals used to generate asynchronous chip select signals are taken from the module's internal bus, which originates in the bus interface unit (BIU) and in the external bus interface (EBI). Each asynchronous chip select has a base address register and option register which contain the programmable characteristics of a particular ACS pin.

There is only one $\overline{DTACK}$ generator in the asynchronous chip select submodule, and it is shared by all of the asynchronous chip selects. The active asynchronous chip select for a particular bus cycle determines the number of wait states produced by the $\overline{DTACK}$ generator before the IMB cycle is terminated.

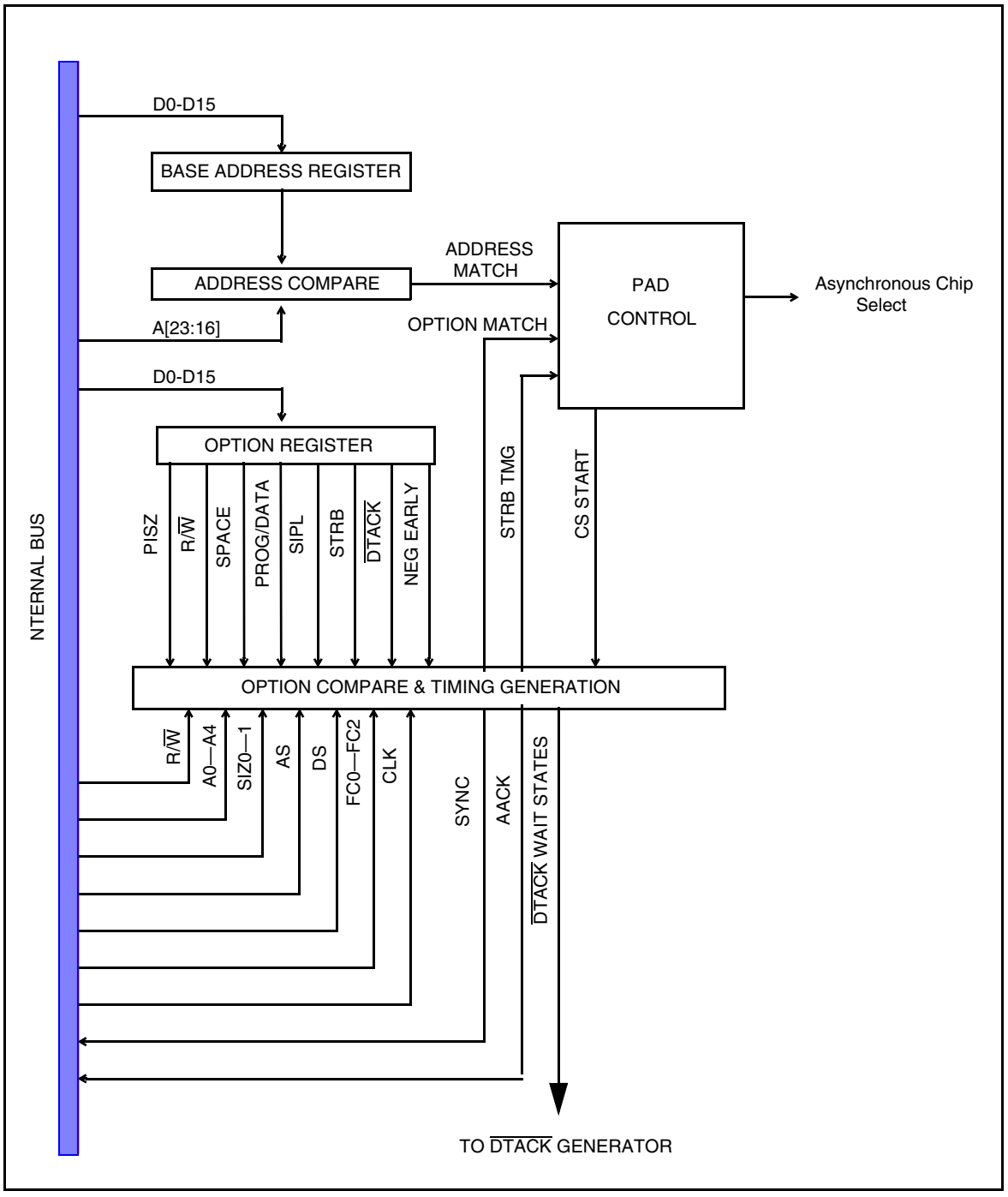


Figure 3-20 Block Diagram of Asynchronous Chip Select

3.6.3 Asynchronous Chip Select Default Attributes

After the MCU resets, the seven asynchronous chip select channels are disabled from participating in the address / option matching process, so that an external device cannot be accidentally selected until an initialization program sets up the base address registers and the option registers.

3.6.4 Asynchronous Chip Select Operations

Each ACS pin can provide a timing signal for an external device and internal bus cycle termination. These functions are enabled independently. To enable the ACS to provide an external timing signal, the ACS function is selected in the corresponding port's pin assignment register. To enable the internal generation of $\overline{\text{DTACK}}$ to terminate an IMB bus cycle, the DTACK field in the asynchronous chip select's option register is set to any value except "external $\overline{\text{DTACK}}$".

Both ACS pin assertion and bus cycle termination depend on an initial address / option match for activation. ACS attributes are defined by its base address register and option register. Each asynchronous chip select participates in the address/option matching process unless the R/$\overline{\text{W}}$ field in its option register is set to "chip select match disabled".

During the matching process, the programmed number of bits from the base address register for each ACS pin are compared against the corresponding internal address bits to determine whether an address match has occurred. This match is further qualified by a comparison of the internal read/write signal, space type, and access size with the programmed values in each asynchronous chip select's option register. When the address and option information match the current cycle, the ACS pin is asserted. If no ACS matches the bus cycle information for an external access, the access is assumed to be to a device with a 16-bit data interface.

More than one asynchronous chip select can be active for a bus cycle. However there is limited interaction between the DTACK field and the PSIZE field of all active ACS pins. If the same number of wait states are selected by each active ACS, the cycle terminates in the expected number of wait states. If the number of wait states differ amoung asynchronous chip selects, the cycle termination operation is undefined. If the size field for any active asynchronous chip select(s) specifies an 8-bit device, the data transfer will be an 8-bit transfer on D[15:8]. If the programmed PSIZE of any two active asynchronous chip selects are inconsistent (a mix of 8- and 16-bit programming), ACS operation is undefined.

The pin corresponding to the active asynchronous chip select may assert for external data transfers with various timing as shown in **Table 3-78**. The pin can be asserted in S0/S1 ($\overline{\text{CE}}$ timing) or in the same state that $\overline{\text{DS}}$ asserts, as selected by the STRB bit in its option register. By selecting appropriate options, an asynchronous chip select can produce a read strobe, write strobe, or output enable for an external device. For example, "Read/Write with CE timing" can be used for a chip enable ($\overline{\text{CE}}$) control, "Read only synchronized with CE" can be used for a read strobe, "Read only synchronized with $\overline{\text{DS}}$" can be used for an output enable ($\overline{\text{OE}}$), and "Write only synchronized with $\overline{\text{DS}}$" can be used for a write strobe.

NOTE

$\overline{CE}$ timing and $\overline{DS}$ timing is different on a read cycle. $\overline{CE}$ timing may assert the pin as early as S0 without regard to internal cycle indication (IAACK). While the $\overline{DS}$ timing option always asserts the pin in S1 with IAACK included in the assertion equation.

To support external devices with synchronous $\overline{OE}$ requirements or slow bus interface logic, the asynchronous chip select pin may be negated one clock before the bus cycle is terminated by the DTACK generator (see **Table 3-81**). This feature provides ample time for a slow bus interface device (during write cycles) to disconnect itself from the current bus cycle before another bus cycle starts.

When no ACS pin is available or when the digital I/O function is selected for the ACS pin, the active asynchronous chip select(s) can still terminate external cycles based on its DTACK value. After the programmed number of wait states expires, DTACK is asserted internally to terminate the IMB bus cycle. If the external $\overline{DTACK}$ pin is asserted before DTACK is asserted internally, the bus cycle is terminated by the external $\overline{DTACK}$.

Table 3-78 ACS Timing Summary

Address	CS Option Register Encodings				CS Timing	
	STRB	DTACK	Negate Early	R/$\overline{W}$	Assert	Negate
SLAM: internal or external	CE	X	X	X	B2	B4
SLAM: internal or external	DS	X	X	X	B2	B4
Internal	CE	X	X	X	B2	B4
External	CE	0	X	X	S1	S5
External	CE	1...14	0	X	S1	S5
External	CE	1...14	1	X	S1	Last S3
External	CE	**External**	X	X	S1	S5
Internal	DS	X	X	X	No assertion	
External	DS	0	X	R	S1	S5
External	DS	1...14	0	R	S1	S5
External	DS	1...14	1	R	S1	Last S3
External	DS	**External**	X	R	S1	S5
External	DS	0	X	W	No assertion	
External	DS	1...14	0	W	S3	S5
External	DS	1	1	W	NOT DEFINED	

Table 3-78 ACS Timing Summary (Continued)

Address	CS Option Register Encodings				CS Timing	
	STRB	DTACK	Negate Early	R/$\overline{W}$	Assert	Negate
External	DS	2...14	1	W	S3	Last S3
External	DS	External	X	W	S3	S5

3.6.4.1 CPU Space Cycle Selection

The asynchronous chip select can also be programmed to identify an IACK cycle, LPSTOP cycle, or breakpoint acknowledge cycle. Table 3-79 shows the pertinent information for programming a ACS to match a CPU space cycle. Bits [15:8] in the base address field must be programmed to match the cycle address bits and the block size must be programmed to 64 Kbytes to allow the address comparator to check the CPU space type on address lines A[19:16]. When the SPACE field is set to “CPU space”, the PROG/DATA field is ignored by ACS logic. The R/$\overline{W}$ field must be set for the particular cycle. The correct setting of PSIZ for an LPSTOP or breakpoint cycle is system dependent. For an IACK cycle, PSIZ is set to “8-bit” to fetch a vector from D15-D8, and PSIZ is set to “16-bit, lower byte” or “16-bit, both bytes” to fetch a vector from D7-D0. For an interrupt acknowledge cycle, the ACS can also check the interrupt priority level, which is given on A[3:1]. A specific interrupt level or IPL (equal to the CS number) is preassigned to each asynchronous chip select. When IPL matching is enabled, the asynchronous chip select will only assert for IACK cycles that match its specific interrupt level. See Table 3-79.

Table 3-79 CPU Space Cycle Information

Cycle Type	FC[2:0]	A[23:20]	A[19:16]	R/$\overline{W}$
IACK	111	1111	1111	read
LPSTOP	111	0000	0011	write
Breakpoint	111	0000	0000	read

3.6.4.2 Chip Select Module Disable

When the STOPCS bit in the BIM module disable register (MDR) is set, the chip select clocks are disabled to save power.

3.6.5 ACS Register Definitions

The asynchronous chip selects are configured by individual base address registers and option registers. These registers are read/write from the IMB. Read operations on the unused bits return zeros and writes to these bits do not have any effect.

3.6.5.1 ACS Address Base Registers (CSBAR1 — CSBAR7)

The base address register consists of a base address field and a block size field. the base address is the starting address for a block enabled by the given asynchronous chip select. The block size determines the extent of the address space from its base

address. Each asynchronous chip select has its own individual base address register, so the address map can be efficiently assigned in any application.

CSBAR1 — Base Address Register **0xYF FA60**
CSBAR7 — Base Address Register **0xYF FA78**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
A23	A22	A21	A20	A19	A18	A17	A16	RESERVED					BLOCK SIZE		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-80 CSBAR1—CSBAR7 Bit Descriptions

Bit(s)	Name	Description
15:8	A[23:16]	Base address fields. The base address field is contained in bits [15:8] of each base address register. Bits [15:8] are used to set the starting address of a particular address space. The address compare logic uses only the most significant address bits to cause an address match within its block size, thus the value of the base address may be selected from any multiple of the block size. **EXAMPLE**: Assuming a block size of 64 Kbytes, the comparator will compare address lines A[23:16] to bits [15:8] of the base register. Therefore, each 64-Kbyte block starts on a 64-Kbyte boundary in the address map.
6:3	—	Reserved
2:0	BLOCK SIZE	Block size fields. Block size is specified by bits [2:0] of each base address register. This field is used to determine the size of the address space enabled by the asynchronous chip select. For decoding a CPU space cycle, A[19:16] must be compared, so the block size must be programmed to 64 Kbytes. 000 = 64 Kbytes, address bits compared = A16 001 = 128 Kbytes, address bits compared = A17 010 = 256 Kbytes, address bits compared = A18 011 = 512 Kbytes, address bits compared = A19 100 = 1 Mbytes, address bits compared = A20 101 = 2 Mbytes, address bits compared = A21 110 = 4 Mbytes, address bits compared = A22 111 = 8 Mbytes, address bits compared = A23

3.6.5.2 ACS Option Registers (CSOR1 — CSOR7)

Each option register consists of individual fields, which determine the timing and the conditions for asserting the asynchronous chip select signals. Since the option register can be used to generate a number of different types of signals required to interface with external devices, care must be taken to program it properly for the desired access type.

CSOR1 — Asynchronous Chip Selection Option Register **0xYF FA62**
CSOR7 — Asynchronous Chip Selection Option Register **0xYF FA7A**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
PSIZ		R/$\overline{W}$		SPACE		PROG/DATA[1]		NEGATE EARLY	RESERVED	SIPL	STRB	DTACK (WAITS) (WAITS)			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NOTES:
1. The PROG/DATA field is ignored if the SPACE field is set to CPU space encoding.

Table 3-81 CSOR1 — CSOR7 Bit Descriptions

Bit(s)	Name	Description
15:14	PSIZ	Port size. The PSIZ option field defines whether the external device supported by the asynchronous chip select is an 8-bit or 16-bit. Additionally, if the device is 16 bits, PSIZ defines whether the asynchronous chip select decodes the upper byte, lower byte or both bytes of the data bus. When a asynchronous chip select is programmed for an 8-bit device, the external device must be connected to D[15:8]. The asynchronous chip select asserts for all addresses which produce an address/option match without regard to whether the transfer is to the upper/lower byte of the data bus. For 16-bit accesses, the BIM initiates multiple bus cycles and provides byte multiplexing of data as needed to satisfy the CPU data transfer. For an access of a 16-bit devicet, the PSIZ field controls the asynchronous chip select pin assertion to properly handle the bus transfer. The asynchronous chip select logic always recognizes which byte(s) is currently selected by the CPU by checking the internal SIZE0-SIZE1 and A0 signals. When a device is 16-bits, PSIZ is programmed to $3 to select the external device for both upper and lower byte transfers. However, when a 16-bit device is implemented by two 8- bit devices, separate enables are required for upper and lower bytes. In this case, two asynchronous chip selects can be programmed as controlling 16-bits. Then one asynchronous chip select is programmed for lower byte (0x1) and the other asynchronous chip select is programmed for upper byte ($2) to enable the correct external device for a byte access. If a MOVEP (Move Peripheral) instruction is used (CPU32 family) to access an 8-bit peripheral, the PSIZ bits in the option register must be set to 16-bit, and either the upper byte or lower byte must be selected according to the peripheral's connection to the data bus. 00 = 8-bit device 01 = 16-bit device, lower byte 10 = 16-bit device, upper byte 11 = 16-bit device, both bytes
13:12	R/$\overline{W}$	Read/write. This option determines whether a asynchronous chip select matches only read bus cycles, only write bus cycles, or both read and write bus cycles. This option field in combination with the STRB field allows the user to generate a variety of control signals for external devices. This option field must be programmed to a non-zero value to allow the corresponding asynchronous chip select to participate in the address/option matching process. When all zeroes are programmed, the asynchronous chip select logic is effectively disabled. The associated pin is driven high, and $\overline{DTACK}$ cannot be asserted internally by that asynchronous chip select. 00 = CS disabled 01 = Read only 10 = Write only 11 = Both

Table 3-81 CSOR1 — CSOR7 Bit Descriptions (Continued)

Bit(s)	Name	Description
11:10	SPACE	Address space. The SPACE option field is compared with the address space indicated by the function codes generated by the CPU during each bus cycle. There are four options which the address space field can select: CPU space, user space, supervisor space and supervisor/user space. If the user application needs to distinguish a program or data space access in supervisor or user space, the PROG/DATA bits can be used. Asynchronous chip select logic recognizes the function codes $0-3 as user spaces, function codes 0x4-6 as supervisor spaces, and function code $7 as CPU space. If the MOVES instruction generates an undefined function code, the space field match is determined by the preceding rule. 00 = CPU space 01 = User space 10 = Supv space 11 = S/U space
9:8	PROG/DATA	Program/data space. This field specifies how to decode program/data space accesses (FC[1] matching) when the SPACE field is not set to the CPU SPACE encoding. If the program/data field is set to 0x11, the operation of the CS match logic is undefined. This field is ignored when the SPACE field is programmed for CPU space. 00 = Data or program 01 = Data space 10 = Prog space 11 = Reserved
7	NEGATE EARLY	Negate CS one clock before cylce terminates. Devices with slow bus interfaces may require extra time to deselect from the current bus cycle due to the device's synchronous $\overline{OE}$ response time or the need for increased data hold time on write cycles. When NEGATE EARLY is cleared the CS pin will negate in S5. When NEGATE EARLY is set, the effect on the CS pin is specified by **Table 3-78**. If cycle termination via the DTACK generator is overridden by the assertion of any internal or external source of DTACK, the CS pin will negate in S5. EXAMPLE: If device access time requires one wait state and the device bus interface is slow, program the DTACK field for two wait states and set the NEGATE EARLY bit. 0 = Negate CS in S5 1 = Negate CS 1 clock before cycle ends
6	—	Reserved
5	SIPL	Specific interrupt level. When an asynchronous chip select is programmed to respond to IACK cycles based on its base address register and its option register encodings, the SIPL option bit determines whether the interrupt priority level (IPL) is compared with the asynchronous chip select's assigned IPL as part of the address/option matching process. Since each asynchronous chip select is assigned to a specific interrupt level, the user must choose the asynchronous chip select with the desired IPL to match IACK cycles. When SIPL is set to one, the assigned interrupt level is compared with the encoded level on A3-A1. Then the asynchronous chip select is only asserted if the IPL matches. If IPL checking is not desired, SIPL is set to zero. In this case the asynchronous chip select will assert on all IACK cycles. 0 = No IPL checking 1 = Only match assigned IPL

Table 3-81 CSOR1 — CSOR7 Bit Descriptions (Continued)

Bit(s)	Name	Description
4	STRB	Strobe timing. This option determines the asynchronous chip select pin assertion state if a match occurs. On a zero wait state write cycle (two clock) the external cycle does not include state S3, therefore neither $\overline{DS}$ nor any asynchronous chip select programmed with STRB = 1 will assert. 0 = Pin asserts as quickly as possible (may assert as early as mid S0) to control device $\overline{CE}$ pins 1= Pin asserts in the same external state as $\overline{DS}$ (read = S1, write = S3)
3:0	DTACK (WAITS)	The DTACK option field determines whether $\overline{DTACK}$ is internally generated. Since DTACK is asserted internally after the programmed number of wait states expires, the user can adjust the bus timing to accommodate the access speed of any external device. With up to 14 wait states selectable, even slow devices can be interfaced with the MCU. If a asynchronous chip select matches on an internal cycle, the DTACK generator is not enabled to assert IDTACK regardless of the field encoding. 0000 = No wait states 0001 = 1 wait states 0010 = 2 wait states 0011 = 3 wait states 0100 = 4 wait states 0101 = 5 wait states 0110 = 6 wait states 0111 = 7 wait states 1000 = 8 wait states 1001 = 9 wait states 1010 = 10 wait states 1011 = 11 wait states 1100 = 12 wait states 1101 = 13 wait states 1110 = 14 wait states 1111 = No internal $\overline{DTACK}$ generation

3.7 Burst Chip Select (BCS)

The burst chip select (BCS) facilitates interfacing industry standard NVRAM and SRAM memory to the BIM without the use of additional glue logic. The BCS is designed to provide improved system performance by the use of burst data transfers to pre-fetch instructions or to support an on-chip cache controller.

3.7.1 Feature List

The following is a list of the major features of the burst chip select.

- Boot chip select — The BCS is the active chip select coming out of reset and can boot from 8- or 16-bit memories.
- Supports industry standard 8- and 16-bit asynchronous memories, 8- and 16-bit synchronous memories, and 16-bit burst mode memories using three standard burst mode protocols.
- Provides better overall system performance with slower memories than previous integration modules.
- The setup and hold times have been improved over the SIM, SCIM, SCIM2, and SLIM which allows faster internal system clock frequencies to support slower external memories.
- High performance chip enable (HP$\overline{CE}$) circuitry provides programmability between high performance performance and low power memory control.
- Programmable burst memory boundary — to support both pre-fetch and cache system architectures, burst data cycles may be terminated at a $2^N - 1$ address

boundary (3, 7, 15, or 31) or after 2^N data transfers (4, 8, 16, 32).

- Bus cycle termination with wait states — the desired number of wait states (to the first data transfer and between burst data transfers) can be programmed by the user. These options control when $\overline{\text{DTACK}}$ and $\overline{\text{BTACK}}$ will be generated internally.
- Programmable block sizes — the memory depth, starting from a programmable base address, can be set to 64 Kbytes, 128 Kbytes, 256 Kbytes, 512 Kbytes, 1 Mbyte, 2 Mbytes, 4 Mbytes, and 8 Mbytes.

3.7.2 BCS and External Memory Configurations

The BCS can be configured to provide the control signals for several different memory configurations. The primary pin set of the BCS consists of five control signals:

- Burst clock (BCLK)
- Load burst address ($\overline{\text{LBA}}$)
- Burst address advance ($\overline{\text{BAA}}$)
- Burst output enable ($\overline{\text{BOE}}$), and
- Burst write enable ($\overline{\text{BWE}}$).

Additionally, some burst RAM configurations require chip select 7 ($\overline{\text{CS7}}$) to provide a $\overline{\text{CE}}$ function.

The set of configurations the BCS supports can be divided up into three categories: one memory, an asynchronous memory, a burst capable memory, and two identical burst capable memories. These are illustrated in **Table 3-82**:

Table 3-82 External Memory Configurations

One Memory	An Asynchronous Memory and a Burst Capable Memory	Two Identical Burst Capable Memories
8-bit/16-bit asynchronous memory	One 8 bit or one 16 bit asynchronous memory and a 16 bit AMCU burst protocol memory	Two 16-bit AMCU burst protocol memories
8-bit/16-bit synchronous memory	One 8 bit or one 16 bit asynchronous memory and a 16 bit pipeline protocol memory	Two 16-bit pipeline protocol memories
16-bit AMCU burst protocol memory		
16-bit pipeline protocol memory		

Table 3-83 lists examples of the types of memories the BCS controls directly. Many other industry standard memories conform to the memory protocols listed below and are compatible with the BCS.

Table 3-83 Directly Supported Memory Memories

Memory	Description	Memory Type	Port K Pins Connect to Memory Pins				
			BCLK	LBA	BAA	BOE	BWE
AM29HL162B (AMD)	1 Mbyte x 16 Sync burst FLASH Burst length = 32	00	CLK	LBA sync 1 CLK	BAA sync	OE async	WE async
28F016XS (Intel)	1 Mbyte x 16 Sync pipeline FLASH Burst length = 32	10	CLK	ADV# sync each BA	CE# sync @S0	OE# async	WE# async 3 CLK
CY7C1032 (Cypress)	64 Kbytes x 16 Sync burst RAM Burst R/W 2-1-1 Burst length = 4	00	BCLK	ADSC sync 1 CLK	ADV sync	OE async	$\overline{WH}$[1] async
MCM67M618 (Motorola)	64 Kbytes x 18 Sync burst RAM Burst R/W 2-1-1 Burst length = 4	00	K	TSC&E sync 1 CLK	BAA sync	G async	$\overline{UW}$[2] async
NM27P6841 (National)	64 Kbytes x 16 Sync burst EPROM Burst length = 4	00	BCLK	TS sync 1 CLK	nc	CS0 async	nc
HN62W5016 (Hitachi)	32 Kbytes x 16 Sync burst MROM Burst length = 8	—	BCLK	—	—	—	—
Industry standard	Async FLASH / EEPROM	11	nc	nc	CE async	OE async	WE async
Industry standard	Async EPROM / ROM / OTP	11	nc	nc	CE async	OE async	nc
Industry standard	Sync RAM (≤20 ns @33 Mhz) Burst R/W 2-1-1	10	CLK	LBA	CE async	OE async	$\overline{WE}_{hi}$ async
Industry standard	Async RAM (>30 ns) Non-burst R/W	11	nc	CE	$\overline{WE}_{lo}$ async	OE async	$\overline{WE}_{hi}$ async

NOTES:
1. An asynchronous chip select (ACS) must be used to control WL for byte write access, $\overline{CS}$ must be controlled by an ACS, external logic, or grounded.
2. An asynchronous chip select (ACS) must be used to control LW for byte write access.

3.7.2.1 Specific Examples:

The NV memory configurations that the BCS supports are shown in Figure 3-21. The BCS provides full programmable control for a single NV burst memory. This memory configuration requires the use of asynchronous chip select (ACS) pins to control the SRAM.

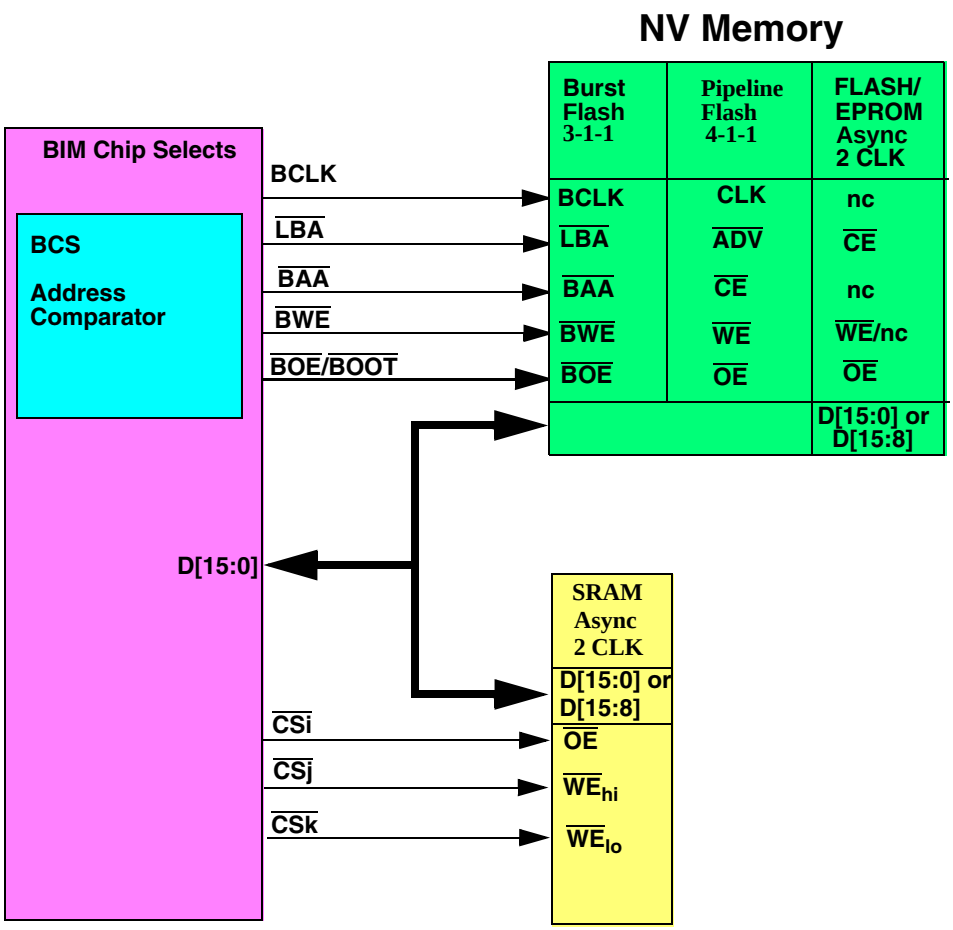


Figure 3-21 Supported NV Memory Configurations

The BCS can be configured to control SRAM memories as shown in **Figure 3-22** provided CS7 is available to control the burst memory CE. In this configuration, the BCS drives the OE of the asynchronous boot [NV] memory through the BOE pin during boot-up configuration, and while code is copied to the SRAM using non-burst accesses. After code is copied to the burst SRAM, and the BCS is programmed to burst on a CS7 address comparator match. Once this is done, the user must guarantee that the code does not try to run in the BCS address range because the BIM will attempt to burst out of the asynchronous memory. By implementing the OEs in this manner, the asynchronous NV memory is used as the boot memory at reset and then the burst SRAM is used after boot configuration to improve system performance.

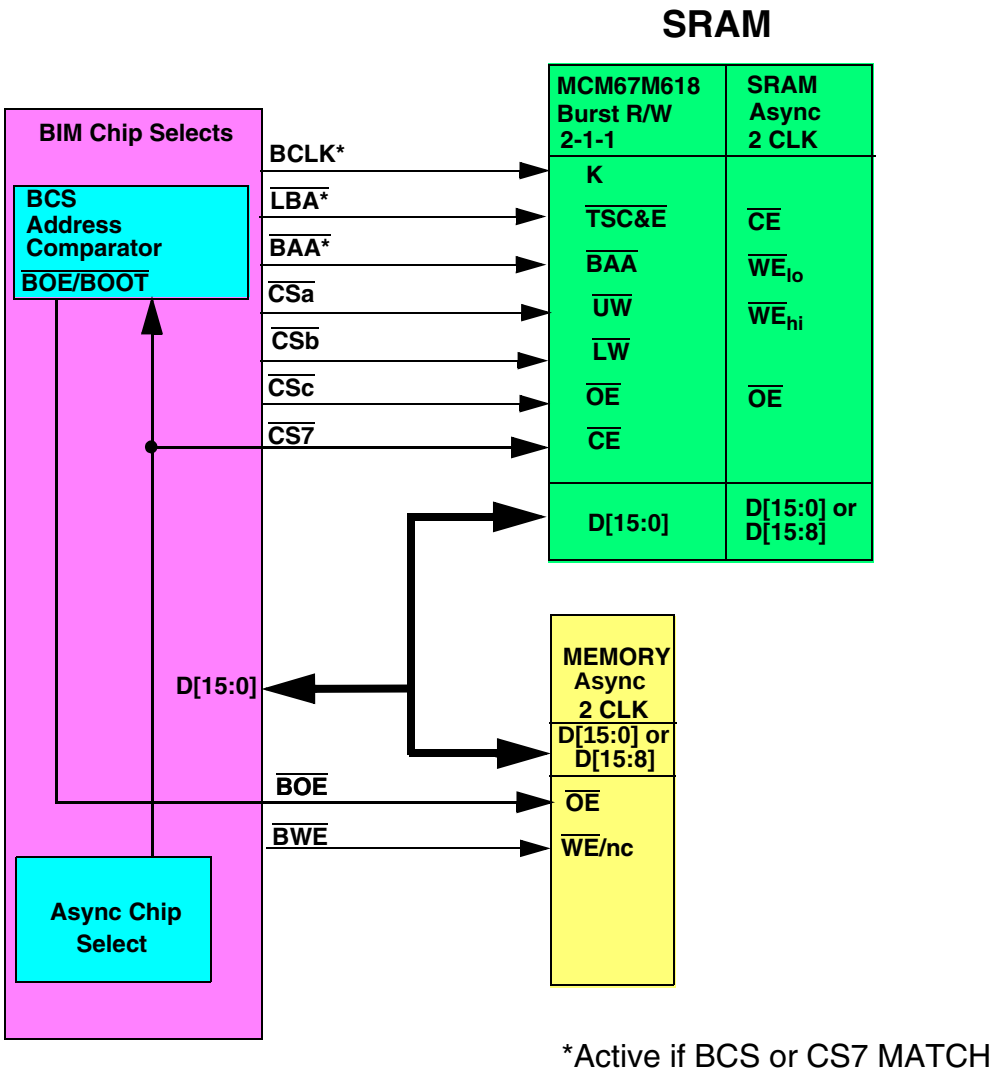


Figure 3-22 SRAM Configurations

The BCS can also be configured to control two memories which use the same burst protocol, as shown in **Figure 3-23**, provided at least three other ASC pins are available. If the NV memory is the boot memory, the chip enable must be tied to ground. If it is not the boot memory, the chip enable can be tied to $\overline{CS7}$ to save power.

In this configuration, the BCS drives the clock, load burst address and burst address advance functions of both memories through the BCLK, $\overline{LBA}$ and $\overline{BAA}$ pins. The $\overline{OE}$ and $\overline{WE_{HI}}$ pins of the burst NV memory are driven through the $\overline{BOE}$ and $\overline{BWE}$ pins. The upper and lower byte write enables and the output enable of the burst SRAM are controlled by the three asynchronous chip select (ACS) pins. The $\overline{CS7}$ address comparator is programmed to match the burst RAM address range for both read and write cycles, and the BCS address comparator is programmed to match the burst flash

address range. The BCS is also programmed to trigger on both BCS and CS7 matches.

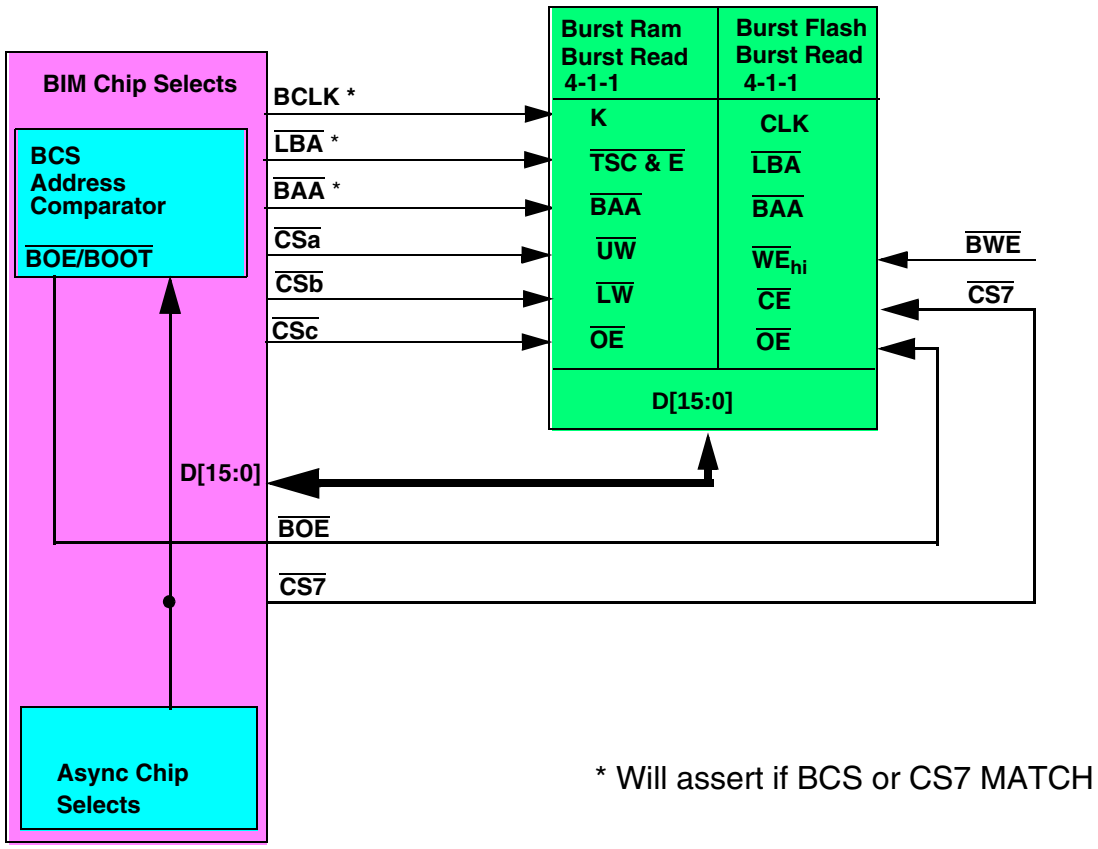


Figure 3-23 BCS Control of Two Like Type Memories, MT = 00

Figure 3-24 shows another two memory memory configuration consisting of an asynchronous NV memory and a non-burst synchronous SRAM. Using the pipelined protocol (MT = 10) in this application, it is possible to achieve high performance 2-1-1 bursts of lengths up to 32 words when using a high speed synchronous SRAM. This performance can be achieved because the BCS asserts LBA on every transfer along with a new address. A burst SRAM could also be utilized in this application, as long as its BAA input is tied high.

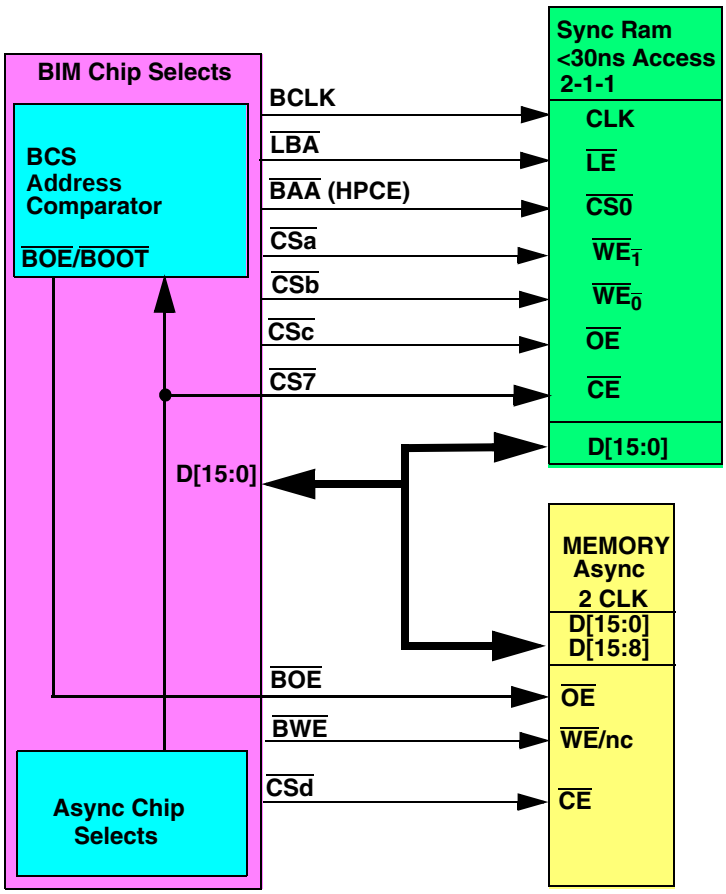


Figure 3-24 Sync SRAM in 2-1-1 Burst Read Application (MT = 10)

3.7.3 BCS Functional Units

The BCS is a full function programmable memory controller consisting of a base address register (BCSBAR), two option registers (BCSOR1, BCSOR2), and control/timing logic. BCSOR1 may be programmed to support

1. Asynchronous RAM, FLASH, and EPROM memories;
2. Burst FLASH, pipeline FLASH, and burst EPROM memories; and
3. Burst SRAM, BCSOR2 configures BCS pin functions.

The BCS contains its own $\overline{\text{DTACK}}$ and $\overline{\text{BTACK}}$ generators. These timing generators allow both the BCS and one standard CS ($\overline{\text{CS1}}$ — $\overline{\text{CS7}}$) to be active concurrently without causing conflicts in the respective $\overline{\text{DTACK}}$ timing generators. The BCS only supports external BTACK in memory type 01, but due to timing constraints, the BCS must be programmed with the same number of initial wait and burst data timing states as the external memory will generate. Any MCU cycle termination event causes the BCS to terminate the current memory access.

The BCS has default reset values in its registers to support CPU reset vector fetches during initialization. At reset, the BCS is configured for non burst read-only operation.

Figure 3-25 shows a block diagram of the BCS. Signals that generate memory control signals are taken from BIM internal buses.

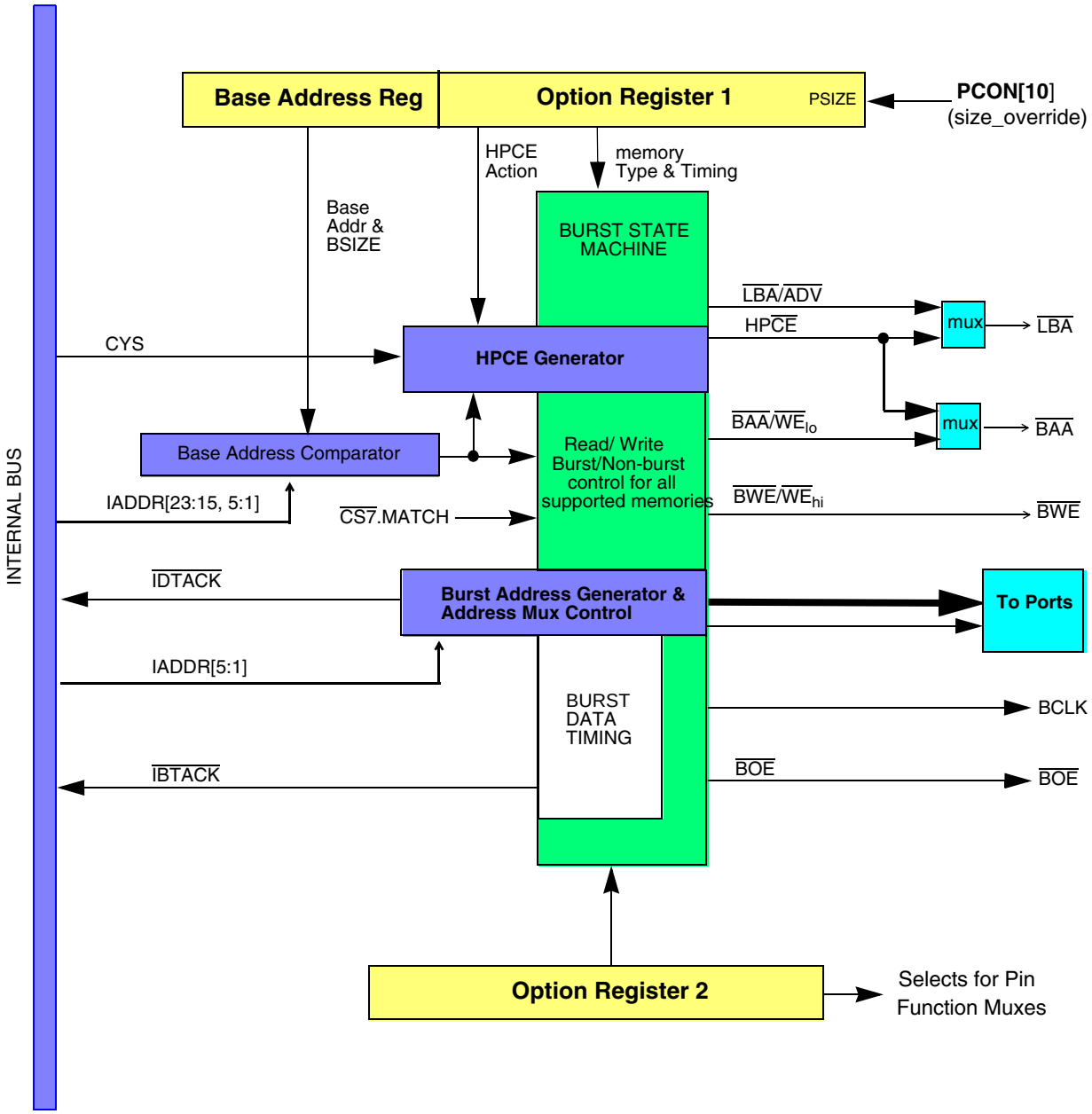


Figure 3-25 BCS Block Diagram

3.7.4 BCS Control Registers

The burst chip select registers are read/write from the IMB. Read operations on unused bits return zeros; writes to these bits do not have any effect. BCSBAR, BCSOR1 and BCSOR2 are reset at SRESET.

3.7.4.1 Base Address Register (BCSBAR)

The base address register consists of a base address field and block size field. The BASE ADDRESS is the starting address of an external memory to be enabled by the burst chip select. The BLOCK SIZE determines the extent of the address space from its base address.

To increase the usable BSC pin strobe window, BCS logic does not include the IAACKB signal (critical timing path) to indicate that an internal module plans to respond to an [internal] cycle. This requires that the address programmed in the base address register and/or the value of the SUPV bit in BCSOR1 be unique among all internal and external devices in the system to avoid executing an external BCS access on an internal cycle.

BCSBAR — Base Address Register **0xYF FA7C**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
BASE ADDRESS								RESERVED					BLOCK SIZE		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-84 BCSBAR Bit Descriptions

Bit(s)	Name	Description
15:8	BASE ADDRESS	Base address. The base address field is contained in bits [15:8] of the base address register. In a supervisor/user space cycle, bits [15:8] are used to set the starting address of a particular address space. Address bits specified A[23:16].
7:3	—	Reserved
2:0	BLOCK SIZE	Block size. The address compare logic uses only the most significant bits to cause an address match within its block size, thus the value of the base address may be selected from any multiple of the block size to configure the address map in the application. **EXAMPLE**: Assuming a block size of 64 Kbytes, the comparator will compare only bits [15:8] of the base register with address lines A[23:16]. Therefore, each 64-Kbyte block starts on a 64-Kbyte boundary in the address map. 000 = 64 KB 001 = 128 KB 010 = 25 6KB 011 = 512 KB 100 = 1 MB 101 = 2 MB 110 = 4 MB 111 = 8 MB

3.7.4.2 BCS Option Register 1 (BCSOR1)

The option register 1 consists of individual fields which determine the timing and the conditions for asserting the BCS signals. Since option register 1 can be used to generate a number of different types of signals required to interface with external memories, care must be taken to program it properly for the desired access type.

A BCS MATCH is defined as a BCSBAR match and a SPACE match.

NOTE

Unlike asynchronous chip select (ACS) operation, memory size (MSIZE) and R/$\overline{W}$ are not option match parameters. Instead, these fields define whether the WE/OE pins(s) assert for the high byte, low byte, both bytes, or no bytes (R/$\overline{W}$ = 0 on a write cycle).

BCSOR1 — BCS Option Register 1 **0xYF FA7E**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
BR		PRO-GRAM SPACE	R/$\overline{W}$	MSIZE	WHOLD	HP$\overline{CE}$		MT		MEMORY BOUNDARY		BDT	INITIAL TIMING		
RESET:															
1	1	1	PCON [10]	PCON [10]	0	1	1	0	0	0	0	1	1	1	1

Table 3-85 BCSOR1 Bit Descriptions

Bit(s)	Name	Description
15:14	BR	Burst response. This field specifies how the BCS will respond to IMB burst cycles that result in a MATCH. The programmable combinations allow the BCS to respond to IMB burst requests by providing full burst read/write operation, burst read/non-burst write operation, and non-burst read/write operation. This field must be programmed to match the characteristics of the memory type (MT) as shown in Table 3-86.
13	PROGRAM SPACE	Supervisor/user address space. The SPACE option is compared with the address space indicated by the function codes during each bus cycle. The burst chip select recognizes function codes 0-3 as user space and function codes 4-6 as supervisor space. The BCS does not respond to any CPU space cycles (FC = 7). When the SPACE bit is cleared, the BCS matches user space accesses only. When the SPACE bit is set, the BCS matches all address spaces except CPU space. 0 = User space. 1 = S/U
12	R/$\overline{W}$	Memory read/write. When cleared, the BCS generates $\overline{BOE}$ on read cycles; when set, the BCS generates both $\overline{BOE}$ and write strobes ($\overline{BWE}$ and $\overline{WE}_{lo}$) as specified by the memory size (MSIZE) and memory type (MT) fields. $\overline{LBA}$ and $\overline{BAA}$ are generated regardless of this setting. 0 = Read only 1 = R/$\overline{W}$
11	MSIZE	Memory size. This bit specifies the memory width. Eight- and 16-bit memory systems are supported in non-burst cycles. **All burst memory types require this field to be set to 16 bits (0)**. $\overline{LBA}$ and $\overline{BAA}$ are generated regardless of this setting. 0 = 16-bit 1 = 8-bit
10	WHOLD	Increase write hold time. On non-burst write cycles ($\overline{BREQ}$ negated or BR field = 1X if the R/$\overline{W}$ field = 1) the BCS will negate the $\overline{BWE}$ and $\overline{WE}_{lo}$ strobes when the initial time period expires. $\overline{DTACK}$ will not be issued for an additional clock to increase write hold time to slave memories. This bit must be set to zero for burst write operation.

Table 3-85 BCSOR1 Bit Descriptions (Continued)

Bit(s)	Name	Description
9:8	HPCE	High performance/low power chip enable. Programming options allow the HPCE to remain asserted for four or eight bus transactions (internal or external) after the last valid BCS memory access, to be negated after every memory access (always forcing an extra wait state to be inserted at the beginning of the next BCS cycle by requiring a valid address/SUPV match before strobe generation), or to always assert. Each time a valid access (MATCH) to the memory is made, the CE time-out period is restarted. The effect of the HPCE logic on the BCS memory state machine is shown in **Figure 3-27**. For highest possible performance when the HPCE state machine is in state CE_ON, the BCS starts a memory access (for MT = 00, MT = 01 or MT = 10) in S0 by asserting LBA without regard to a MATCH on any MCU cycle [internal or external] to maximize the time available for external memory access. In S1, the memory access will be aborted (BOE/BWE pins will not assert) if the address decode and cycle type is not a MATCH. When in the CE_OFF state, a valid BCS address and SUPV match is required before CE asserts, this adds an extra wait state to the initial access time to allow the memory to power-up before an access is started. **Figure 3-27** shows the effect of HPCE on timing. In the first cycle HPCE is initially negated, and then asserted at the start of the following bus cycle. If HPCE is asserted when LPSTOP executes, the HPCE signal is negated to power down any external memory controlled by HPCE and the bus cycle counter is cleared. The HPCE state machine remains in the CE_ON state so that when LPSTOP is exited, HPCE re-asserts and four or eight non-BCS cycles must occur before HPCE negates. 00 = Negate 4 bus cycles after last access 01 = Negate 8 bus cycles after last access 10 = Always negate after cycle 11 = Never negate
7:6	MT	Memory type. These bits specifies the external memory type. The BCS supports burst memories with internal address generation (memory type = 00), burst memory with internal address generation and BTACK generation (memory type = 01), pipeline burst memory (memory type = 10), and asynchronous memory (memory type = 11). The memory state machine for asynchronous memory, and burst memory protocol is shown in **Figure 3-28**. Memory type 01 supports burst memory architectures with no physical boundry limitations (continous burst) that control each data transfer utilizing the memory's internal BTACK generator. BAA is not a required control signal and can function as the HPCE. When the memory transfers the last word in its output buffer, one or more wait states may be needed to reload the output buffer with the next page of data from the [memory] array before contining the burst transfer. In the current version of the BCS, the initial wait, burst data timing and memory boundary fields are still used by the BCS to determine when to terminate the bus cycle by asserting IDTACK. These fields should be set to their maximum values to utilize the longest burst data transfer supported by the BCS. Although the number of data transfers is determined by the memory access time and the wait states needed to reload the output buffer, the BCS supports a maximum burst length of 72 words with a 2-1-1 burst memory. 00 = Burst FLASH and SRAM 01 = Burst FLASH with external BTACK generation 10 = Pipeline FLASH 11 = Async FLASH and SRAM
5:4	MEMORY BOUNDARY	Memory boundary. These bits set the physical memory boundary for the burst address generator. Refer to **3.7.5.6 Burst Address Generator (BAG)** for a complete description of the BAG operation. 00 = 4 01 = 8 10 = 16 11 = 32

Table 3-85 BCSOR1 Bit Descriptions (Continued)

Bit(s)	Name	Description
3	BDT	Burst data timing. After the first word of a burst transaction has been transferred, the burst data timing bit specifies if a wait state is inserted between burst data transfers. If the burst data timing bit is cleared, the BCS does not insert any wait states between burst data transfers; if set, the BCS inserts a single wait state between data transfers (see **Figure 3-34**). If BDT is set and memory type = 00 or 01, the BAA waveform changes. If BDT is set and memory type = 10, the LBA waveform and BAG increment timing change. 0 = No waits between burst data. 1 = 1 wait between burst data
2:0	INITIAL TIMING	Initial access timing/$\overline{\text{DTACK}}$ generation. On non-burst cycles or when the burst response field is set to non-burst operation, the initial access timing field is used to specify the number of wait states to insert in the external cycle before the BCS terminates the cycle (asserts $\overline{\text{DTACK}}$). Setting the WRITE HOLD bit adds one additional wait state to write cycles than the initial timing field specifies. On burst cycles, the initial timing field specifies the amount of wait states that are inserted before the first word of a burst transaction is transferred (first $\overline{\text{BTACK}}$ generated). Since $\overline{\text{DTACK}}$ and $\overline{\text{BTACK}}$ are asserted internally after the programed number of wait states expires, the user can adjust the bus timing to accommodate the access speed of the external memory. With up to seven wait states, slow memories can be interfaced with the MCU. If any MCU event terminates the cycle ($\overline{\text{BERR}}$ or $\overline{\text{DTACK}}$) before the BCS generates $\overline{\text{DTACK}}$, the BCS terminates the memory transfer. The BCS does not allow external $\overline{\text{BTACK}}$ to be asserted during a burst memory transfer in the BCS memory range, except as discussed for MT = 01. 000 = 0 001 = 1 010 = 2 011 = 3 100 = 4 101 = 5 110 = 6 111 = 7

Table 3-86 Legal Burst Response and Memory Type Combinations

Burst Response Field	MT = 00	MT = 01	MT = 10	MT = 11
00	LBA asserts if HPCE is on, BAA, BOE and BWE are disabled.			Disabled
01	Supported operation	Supported operation	Not supported	Not supported
10	Supported operation	Supported operation	Supported operation	Not supported
11	Supported operation	Supported operation	Supported operation	Supported operation

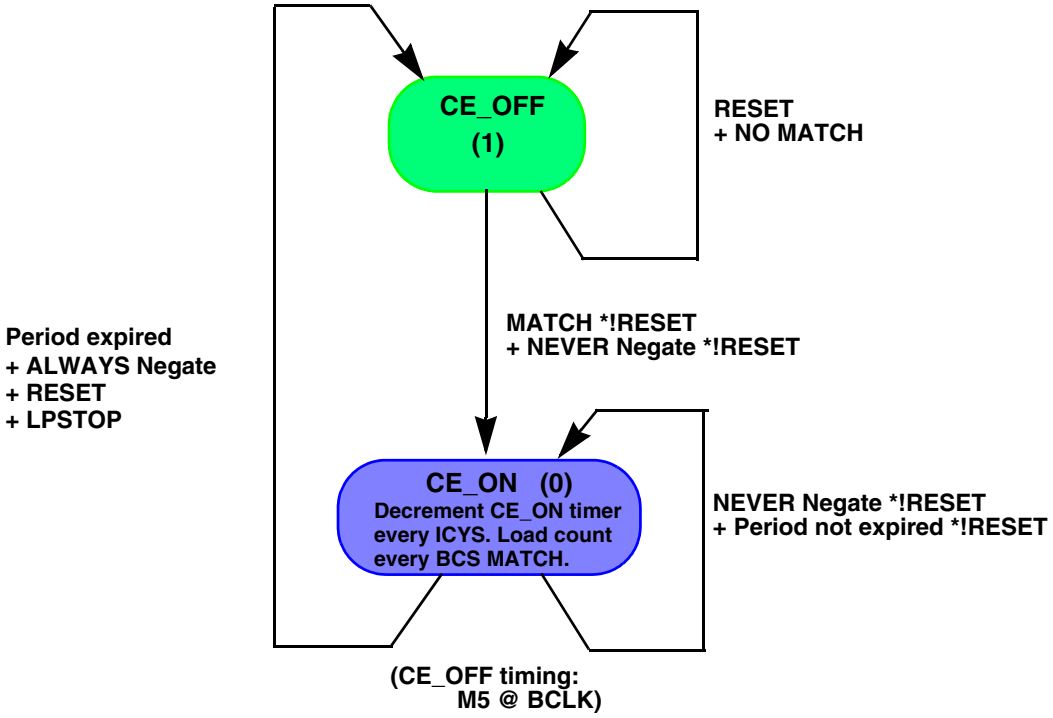


Figure 3-26 HPCE Functional Operation

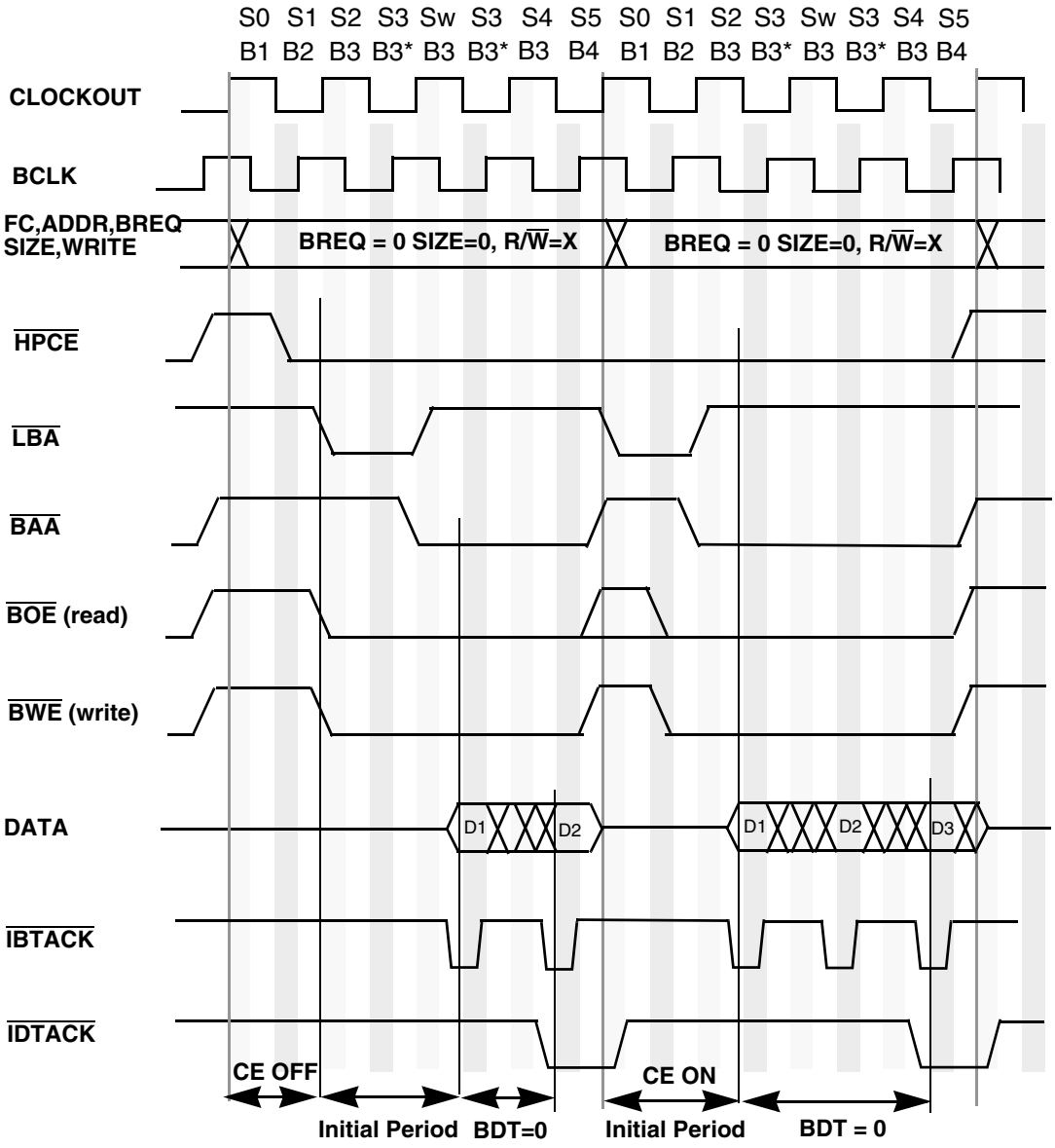


Figure 3-27 Effect of HP$\overline{CE}$ on BCS Operation, Initial Period = 0.

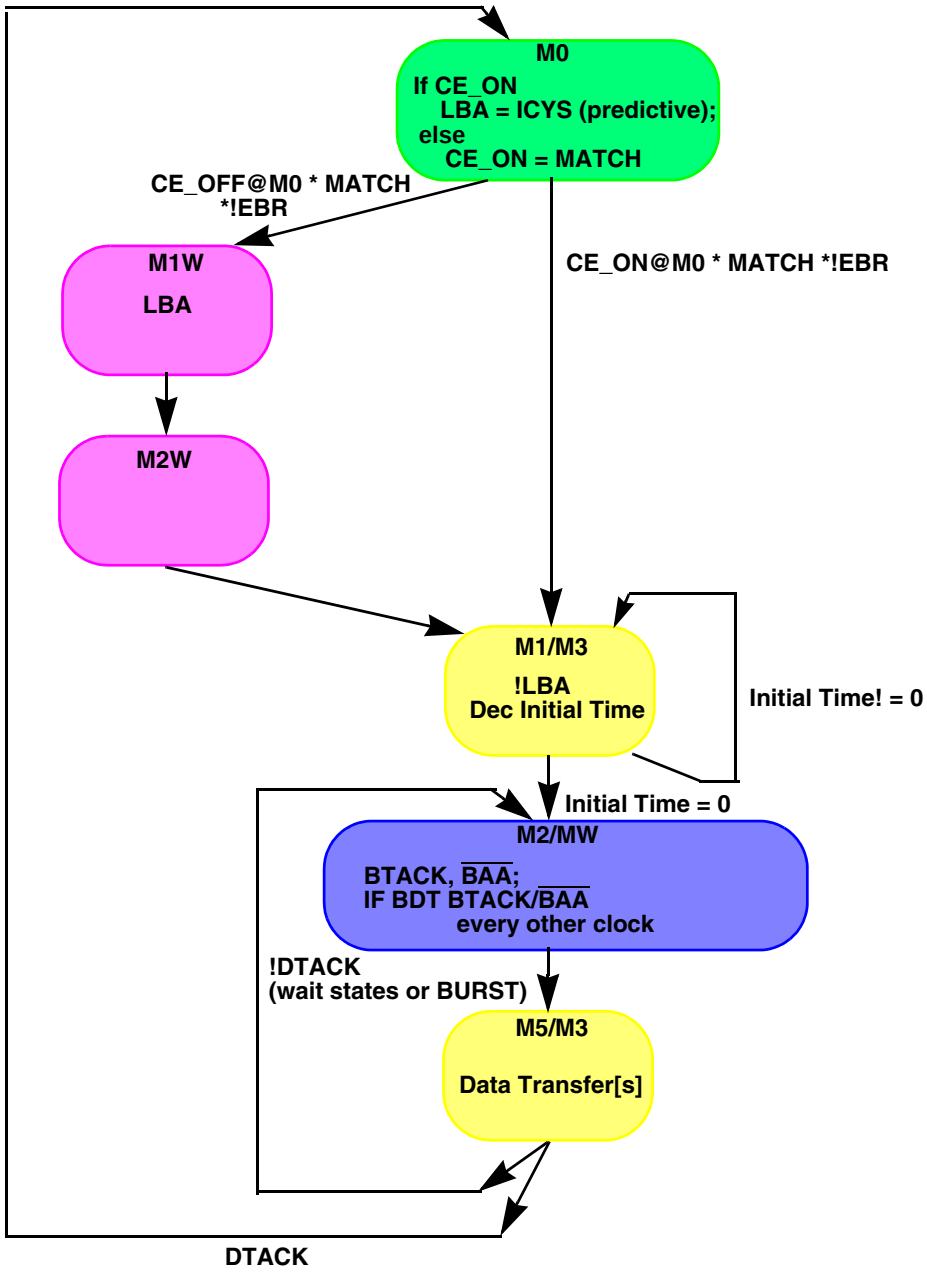


Figure 3-28 BCS State Machine for Burst, Pipeline, and Async Protocols

3.7.5 BCS Operation

3.7.5.1 RESET

During reset, BCLK is active, $\overline{BAA}$ and $\overline{BWE}$ are driven high. Any memory connected to the $\overline{BOE}$ override pin must be disabled (i.e., $\overline{CE}$ negated) during reset to avoid causing bus contention during the reset [PCON override] period.

3.7.5.2 Boot Memory Select

After reset, the burst chip select defaults to non-burst operation (burst response bits in option register 1 = 11). In this default mode the BCS is capable of accessing all supported memories.

In non-burst operation, the BCS supports 8- or 16-bit boot memories. The default memory size (MSIZE) is set by PCON[10] and may be overridden by the SIZE pin during reset. If MSIZE is set to 8 bits, the EBI will execute two [byte] cycles each time the IMB master requests a word transfer. $\overline{\text{LBA}}$ will be asserted during each byte cycle of a word transfer.

3.7.5.3 Non-Burst (Asynchronous) Operation

In this section, a non-burst memory access refers to a single data transfer per bus cycle to a synchronous or asynchronous memory. For synchronous memories, the memory type field in option register 1 (OR1) must be set to either 00, 01, or 10 depending on the specific protocol that is required. Asynchronous memories require the memory type field in OR1 to be programmed as 11. XXX through XXX show non-burst memory accesses for memory types 00, 01 and 10, while XXX shows access to an asynchronous memory (memory type 11).

When the burst response field of OR1 is programmed for non-burst operation, or during non-burst cycles ($\overline{\text{BREQ}}$ is negated), the BCS executes either an 8- or 16-bit non-burst memory access, depending on the programming of the MSIZ bit in OR1. When performing a non-burst memory access, the programming of the initial timing field of OR1 determines the number of wait states to be inserted in the external cycle before the BCS generates $\overline{\text{DTACK}}$. The memory boundary field, burst data timing field and BAG mode field are ignored by BCS logic during a non-burst access.

Figure 3-29 shows a 16-bit non-burst read access to a 16-bit memory with five wait states. If HPCE is on at the start of the memory access $\overline{\text{LBA}}$ will assert during S0 to load the current address into the memory; if HPCE is off an extra clock is added to the access to allow the memory to power up (assert CE). $\overline{\text{BOE}}$ will assert during S1, to enable the memory data drivers, and negates in the second half of S5. $\overline{\text{BAA}}$ and $\overline{\text{BWE}}$ remain negated throughout memory read access.

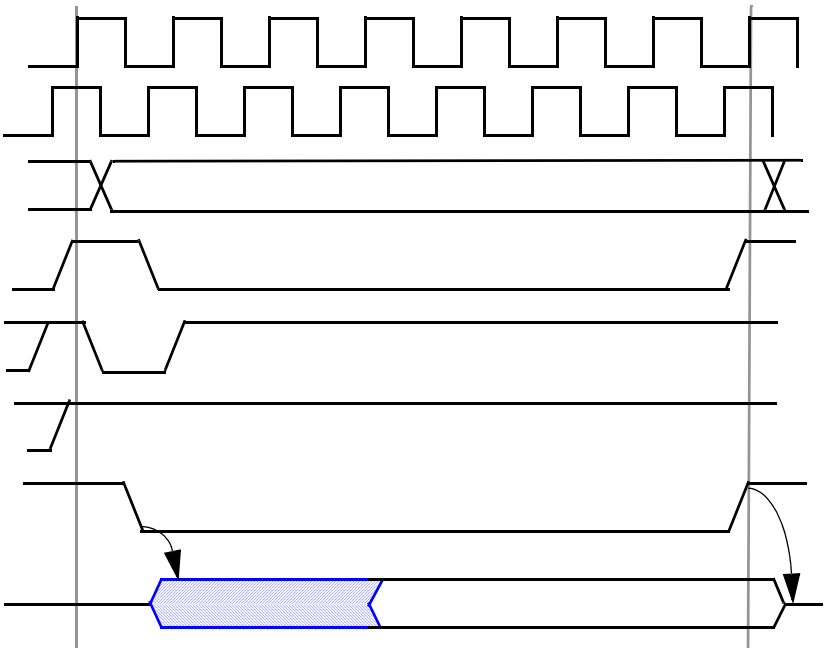


Figure 3-29 Non-Burst Read to 16-Bit Sync Memory

Figure 3-30 shows a 16-bit non-burst write access to a 16 bit memory, with five wait states. If $\overline{\text{HPCE}}$ is on at the start of the memory access, then $\overline{\text{LBA}}$ will assert during S0. $\overline{\text{BWE}}$ will assert during S1, if the write hold bit in OR1 is set to a zero, it will negate during the second half of S5. If the write hold bit is set to a one, an extra clock cycle is added to the number of wait states specified in the initial wait field and $\overline{\text{BWE}}$ will negate one clock before the cycle terminates (S3) to give a slow memory extra address and data hold time on the bus. $\overline{\text{BAA}}$ and $\overline{\text{BOE}}$ remains negated throughout a memory write access.

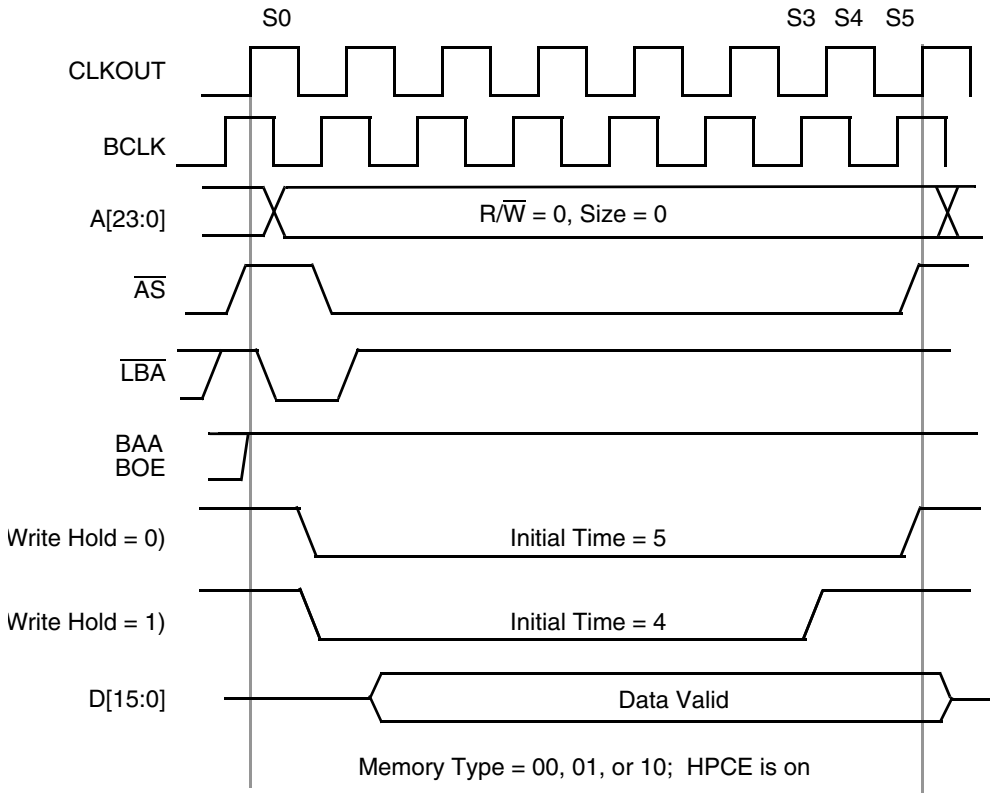
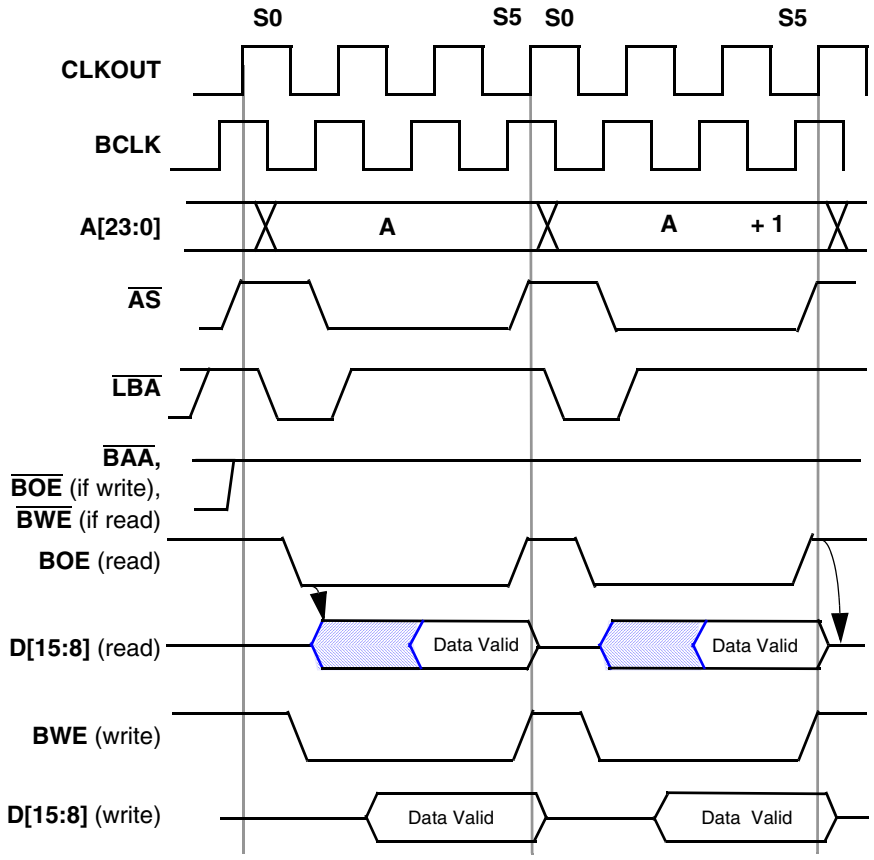


Figure 3-30 Non-Burst Write to 16-Bit Sync Memory

Figure 3-31 shows a 16-bit non-burst read and write access to an 8-bit memory. During the first memory access, A[0] = 0, and during the second A[0] = 1. For both read and write cycles, if HPCE is on at the start of the access, then LBA will assert for one clock during S0 of both byte accesses. During a memory read access, BOE will assert during S1 and will negate mid-S5. BAA and BWE will remain negated throughout the memory read access. During a memory write access, BWE will assert during S1 and will negate in either S3 or S5 depending on the setting of the write hold bit in OR1. BAA and BOE remain negated throughout the memory write cycle.



HPCE is on, Memory Type = 00,01,10, Write Hold = 0, Initial Wait = 1

Figure 3-31 16-Bit Non-Burst Cycle to 8-Bit Memory

When the memory type field in OR1 is set to 11, then the BCS will execute asynchronous read and write cycles. When using the asynchronous protocol to access [asynchronous] memories, $\overline{LBA}$ is not required, and can be used as $\overline{HPCE}$. $\overline{BWE}$ is the write enable for the high byte of a word ($\overline{WE}_{hi}$) and $\overline{BAA}$ functions as the write enable for the low byte of a word ($\overline{WE}_{lo}$).

Figure 3-32 shows a 16-bit asynchronous read followed by an 8-bit asynchronous write to the high byte (A[0] = 0) and an 8-bit asynchronous write to the low byte (A[0] = 1) (MSIZ = 0 in this figure). During a read cycle, $\overline{BOE}$ will assert when transferring data from the high byte, low byte or both bytes. During a write cycle $\overline{BWE}$ and/or $\overline{BAA}$ will assert depending whether the high byte, low byte or both bytes are being transferred. In the first write cycle in **Figure 3-32**, only the high byte of a word is being transferred, so only $\overline{BWE}$ asserts. In the second write cycle, only the low byte of a word is being transferred, so only $\overline{BAA}$ asserts. If both bytes are being transferred, then both $\overline{BWE}$ and $\overline{BAA}$ will assert.

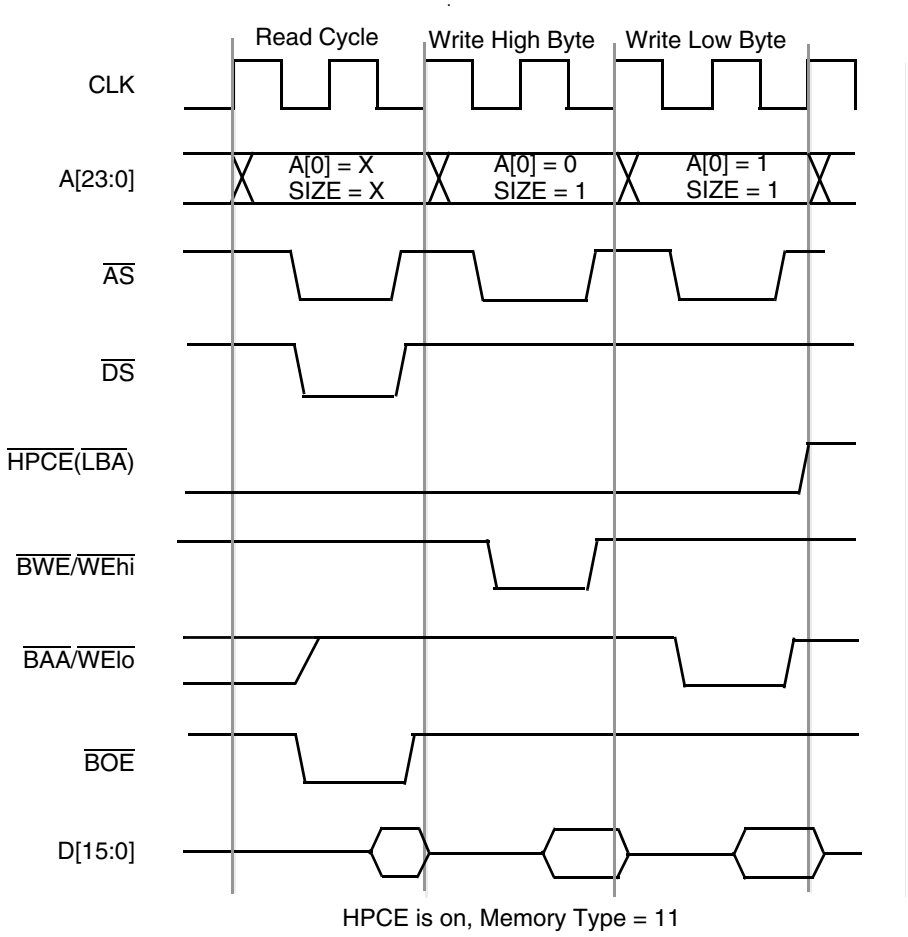


Figure 3-32 2 CLK Read/Write Cycles to Async Memory

3.7.5.4 Burst Operation

When the burst response field is programmed for burst operation and the match condition is satisfied, the BCS executes the memory access protocol as specified by the memory type field in OR1. The following programming restrictions apply for burst operation: the write hold bit must equal zero, MT cannot equal 11, BR must equal 10 or 01, and MSIZE must equal zero.

After waiting the INITIAL ACCESS TIMING wait states, internal $\overline{\text{BTACK}}$ and $\overline{\text{BAA}}$ assert to transfer the first data word between bus master and memory. If the BURST DATA TIMING bit indicates no wait states between data transfers, then internal $\overline{\text{BTACK}}$ and $\overline{\text{BAA}}$ remain asserted throughout the burst cycle; otherwise, internal $\overline{\text{BTACK}}$ and $\overline{\text{BAA}}$ toggle such that a wait state is inserted on each burst data transfer.

3.7.5.5 Burst Termination

The AMCU burst protocol allows a burst cycle to be terminated by either the bus master, or the slave. All burst cycles are terminated by the assertion of $\overline{\text{DTACK}}$ or $\overline{\text{BERR}}$.

DTACK can be generated by the burst address generator (BAG) or any DTACK/BERR source in the MCU.

A slave can terminate the burst cycle on the current data transfer by asserting DTACK if it cannot continue the burst. The IMB master may terminate the burst cycle on the current data transfer by negating BREQ. If the slave inserts wait states on this transfer, the cycle ends when the data is transferred (internal BTACK is asserted).

Depending on the BAG mode, the burst address generator terminates burst cycles based on the burst address or on the number of burst transfers.

3.7.5.6 Burst Address Generator (BAG)

The burst address generator (BAG), consists of two modulo 32 counters and logic to control the burst address mux (in Port A) to output the burst address on the external address bus A[5:1], generate each individual burst address during a burst cycle, and terminate the burst cycle based on the memory's internal architecture requirements.

The burst address (BA[5:1]) is brought out through the external address bus A[5:1] for burst memories requiring an [incremented] burst address for every data transfer. The BAG supports both program pre-fetch and cache line operation.

Table 3-87 BAG Termination Conditions

Memory Type	BAG Address Generation/Tracking Requirements	
	Pre-Fetch Operation	Cache Operation
00 - Burst FLASH / SRAM	Inc internal address@BTACK Terminate Cycle when BA[5:1] = MOD[1:0] value.	Inc modulo counter @BTACK Terminate cycle after MOD[1:0] data transfers.
10 - Pipeline FLASH / SRAM	Inc internal address@BTACK Gen external address@S2 ->A[] Terminate Cycle when BA[5:1] = MOD[1:0] value.	Inc internal address@BTACK Gen external address@S2 -> A[] Inc modulo counter @BTACK Terminate cycle after MOD[1:0] data transfers.
11 - Async FLASH / SRAM	None	None

3.7.5.7 Incrementing the Burst Address

The BAG latches the starting burst address and increments the burst address by one word every time internal BTACK is asserted.

3.7.5.8 BAG Cycle Termination Conditions

The BAG terminates the burst cycle based on the physical address or the number of burst transfers completed.

Table 3-88 BAG Termination Conditions

Memory Boundary	Burst Cycle Termination	
BCSOR1[5:4]	PRE_FETCH Mode	Cache Mode
00 = 4	BA[2:1] = 0x3	After 4 data transfers.
01 = 8	BA[3:1] = 0x7	After 8 data transfers
10 = 16	BA[4:1] = 0xF	After 16 data transfers
11 = 32	BA[5:1] = 0x1F	After 32 data transfers

In pre-fetch mode operation (BAG MODE = 0), if the burst cycle accesses memory location (2^N-1), the BCS terminates the burst by asserting $\overline{\text{DTACK}}$ internally. This action forces the bus master to restart the burst operation on the next bus cycle, starting at address 2^N. For burst memories with no physical boundary restrictions, this field is programmed to %11 to select the maximum burst length supported by the BCS.

In cache mode operation (BAG MODE = 1), the BAG terminates the burst transaction on the 2^N data transfer regardless of the burst address.

3.7.5.9 Burst Waveforms

Figure 3-33 and Figure 3-34 show several different waveforms for the three burst protocols that the BCS supports. These figures show the affect of the programming of the initial timing and BDT fields in OR1 and the HPCE circuit on the waveforms produced on the $\overline{\text{LBA}}$, $\overline{\text{BAA}}$, $\overline{\text{BOE}}$ and $\overline{\text{BWE}}$ pins.

Figure 3-33 through Figure 3-38 apply to both MT = 00 and MT = 01, but the external $\overline{\text{BTACK}}$ assertion at the bottom of these figures only applies to MT = 01. **The assertion of external $\overline{\text{BTACK}}$ in memory types 00, 10 or 11 causes abnormal BCS operation.**

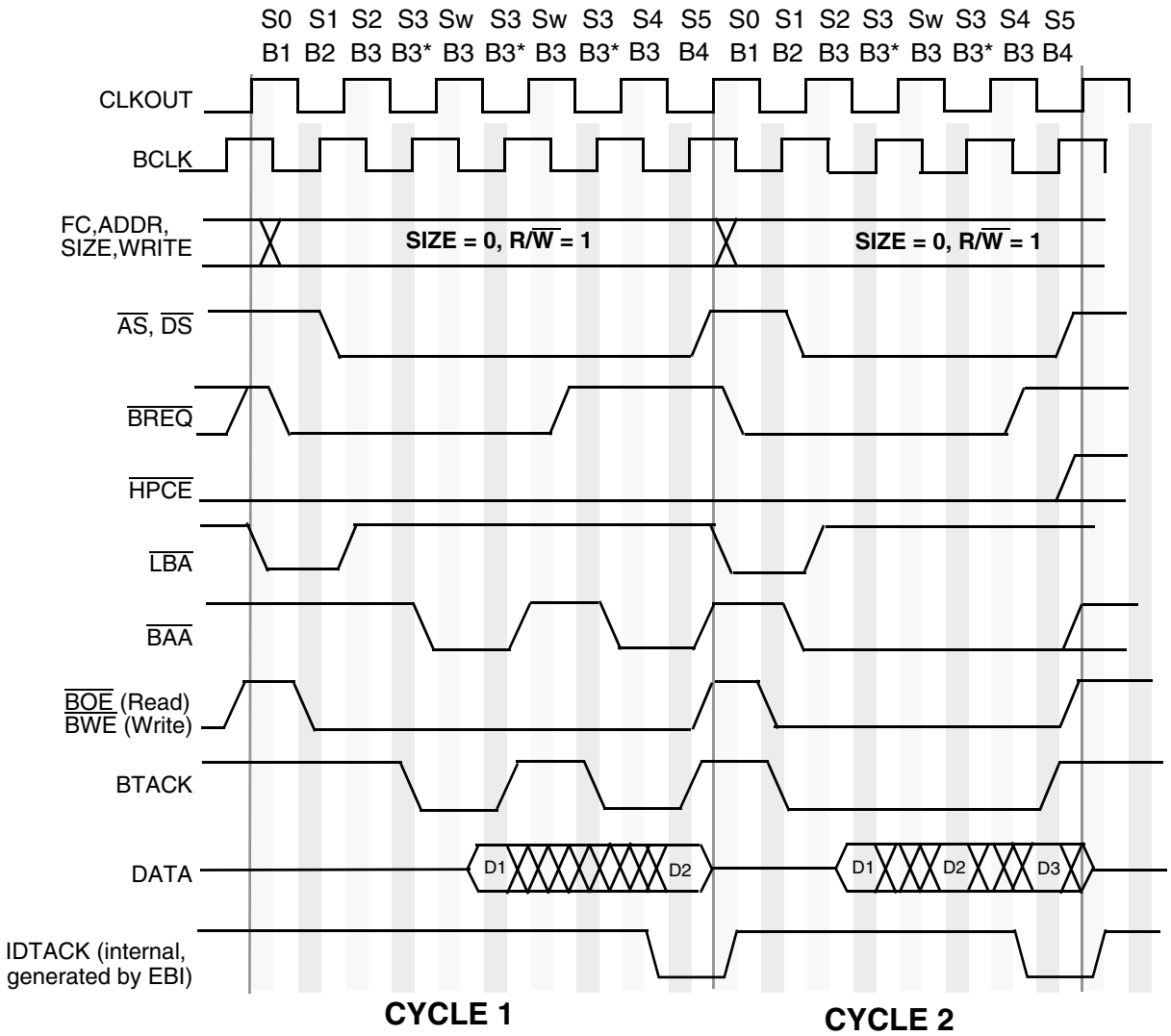
Figure 3-33 demonstrates the burst read protocol assuming that the $\overline{\text{HPCE}}$ circuit is on at the start of cycle #1. Cycle #1 also assumes both the initial wait and the BDT fields in OR1 were programmed to one, thus inserting one wait state in the initial access time, and one wait state between data transfers. This is known as a 3-2-2 burst cycle. Cycle #2 assumes that both the initial wait and the BDT fields in OR1 were programmed to zero, thus producing a burst read cycle with no wait states. This is know as a 2-1-1 burst cycle.

When $\overline{\text{HPCE}}$ is on at the start of a cycle, then $\overline{\text{LBA}}$ asserts by mid-S0 and negates one clock later. When the BAG counts the number of clocks programmed in the initial wait field of OR1, it will generate IBTACK and $\overline{\text{BAA}}$ will be asserted. If BDT = 1, then $\overline{\text{BAA}}$ will negate 1 clock after it asserts, as in cycle 1. If BDT = 0, then $\overline{\text{BAA}}$ remains asserted through the entire cycle, as in cycle 2. When $\overline{\text{BREQ}}$ negates during a burst cycle, the EBI terminates the bus cycle on the next data transfer (i.e., BTACK assertion). This is known as master terminated burst. In master terminated burst cycles, EBI logic is responsible for asserting IDTACK when BTACK asserts.

The waveform for $\overline{\text{BTACK}}$ at the bottom of shows how the external memory would need to assert and negate the $\overline{\text{BTACK}}$ pin in MT = 01 to produce the same waveforms on $\overline{\text{BAA}}$ and IBTACK as the BAG produces in MT = 00.

Figure 3-33 demonstrates burst read protocol with HPCE off at the start of cycle 1. In cycle 1 the initial wait field is programmed as one and the BDT field is programmed to 0, thus inserting one wait state in the initial access time, and no wait states between data transfers. This is known as a 3-1-1 burst cycle. In cycle 2 both the Initial Wait and the BDT fields are zero.

If $\overline{\text{HPCE}}$ is off at the start of a bus cycle, the BCS will assert $\overline{\text{HPCE}}$, and will wait one clock before asserting $\overline{\text{LBA}}$. This is to allow time for the external memory to power-up before the address is loaded into it. In comparing the assertion of $\overline{\text{LBA}}$ in cycle 1, to cycle 1, note that $\overline{\text{LBA}}$ asserts in the same clock state as $\overline{\text{AS}}$ instead of 1 clock before $\overline{\text{AS}}$. Once $\overline{\text{LBA}}$ asserts, the rest of the bus cycle progresses as described above. Since HPCE does not negate at the end of cycle 1, cycle 2 executes the same as cycle 2 in .



Cycle1: HPCE is on, Initial Timing = 1, BDT = 1 Cycle2: Initial Timing = 0, BDT = 0

NOTE: If the master terminates the burst cycle, the BIM asserts internal DTACK (IDTACK) when the [last] BTACK asserts.

Figure 3-33 Burst Read Protocol, (MT = 00 or 01)

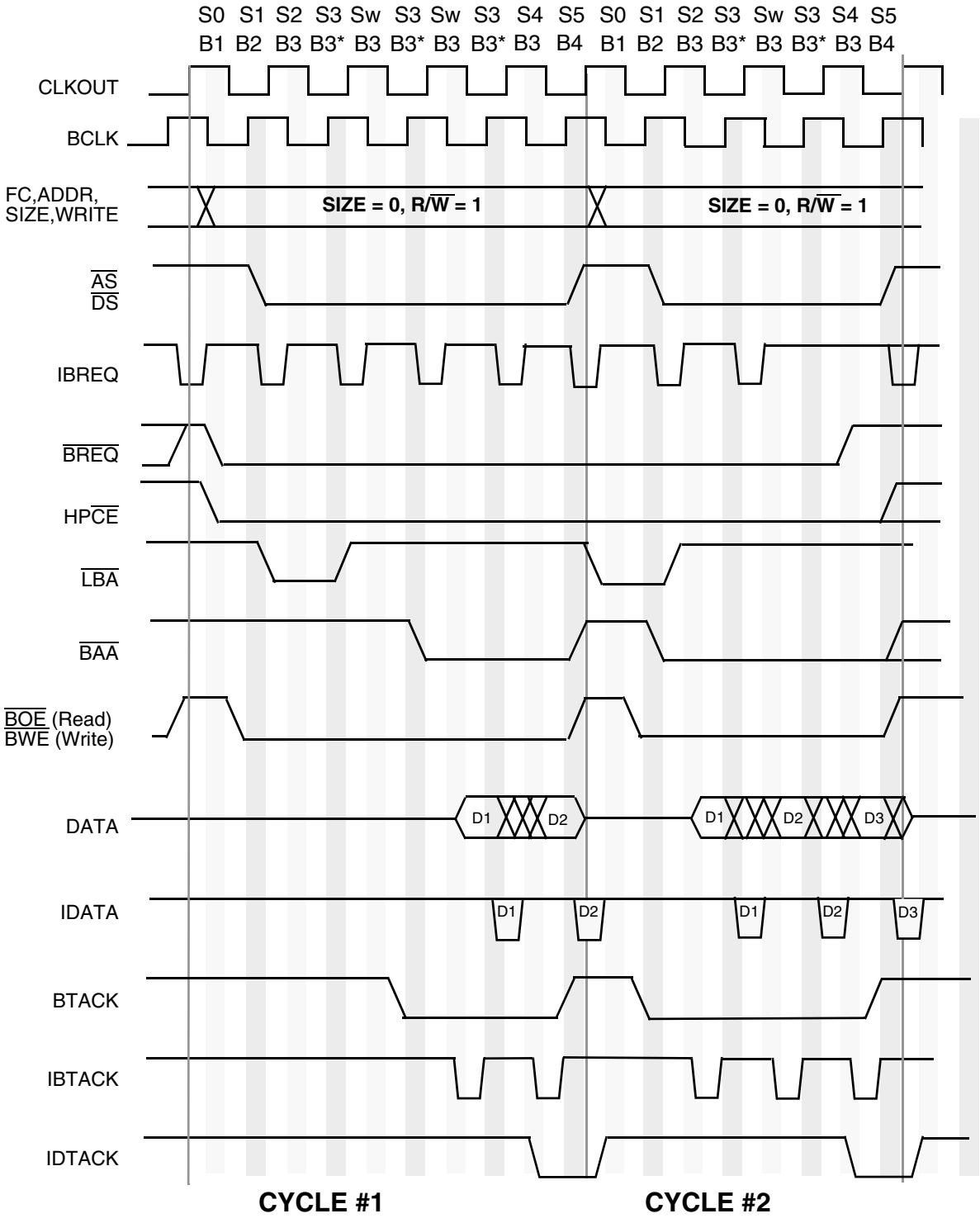


Figure 3-34 Burst Read Protocol, (MT = 00 or 01)

Cycle #1: Initial Timing = 1, BDT = 0, HPCE is initially off
Cycle #2: Initial Timing = 0, BDT = 0

The pipelined protocol (MT = 10) is intended for external memories which must load a new address for each beat of data it is to provide. See **Figure 3-35** for burst waveforms for the pipelined protocol (MT = 10).

Figure 3-35 demonstrates the pipelined protocol, assuming that the $\overline{\text{HPCE}}$ circuit is on at the start of cycle #1. Cycle #1 also assumes both the initial wait and the BDT fields in OR1 were programmed to one, thus inserting one wait state in the initial access time, and one between beats of data. Cycle #2 assumes that both the Initial Wait and the BDT fields in OR1 were programmed to zero, thus producing a burst read cycle with no wait states.

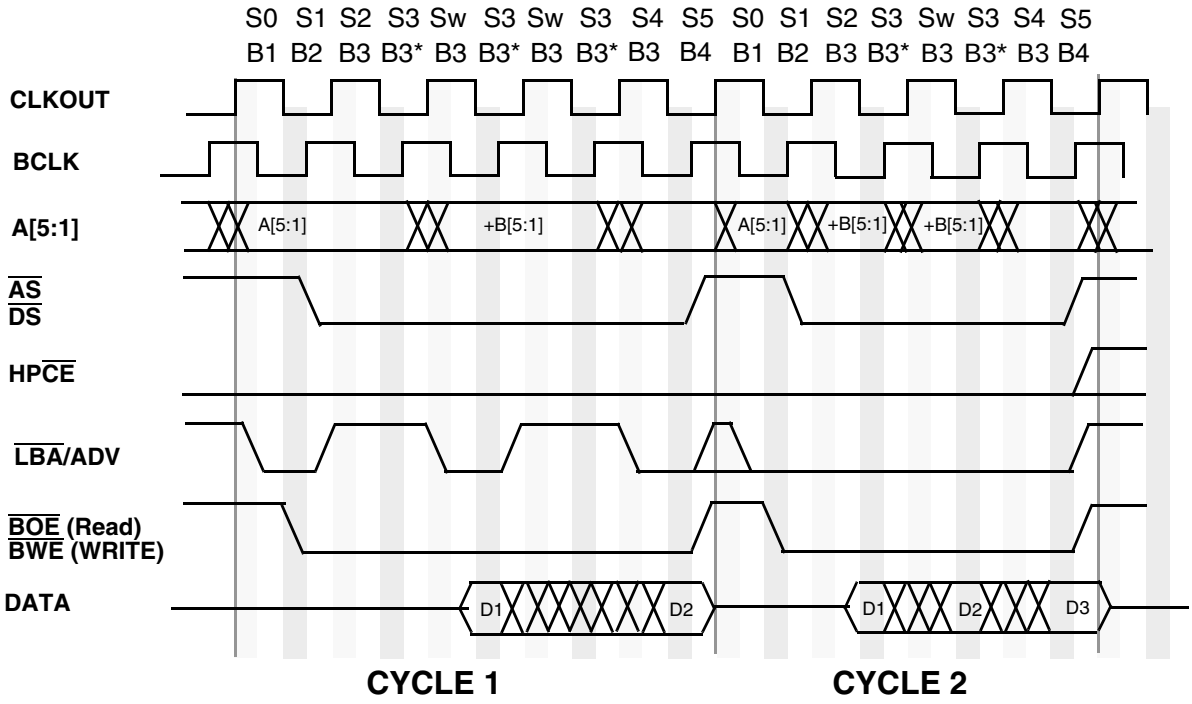
In cycle #1, $\overline{\text{LBA}}$ asserts at the start of the cycle to load the initial address into the external memory. Since BDT = 1 in this case, $\overline{\text{LBA}}$ will negate one clock later to prevent a new address from being loaded into the external memory on the next rising edge of BCLK. With BDT = 1, $\overline{\text{LBA}}$ will toggle every BCLK until the cycle terminates. Also note that when BDT = 1, the BAG will increment A[5:1] every other BCLK, instead of every BCLK, as demonstrated in cycle #2 (where BDT = 0). In pipelined mode, the BAG still counts the Initial Timing number of clocks before generating IBTACK, and will add one wait state between beats of data if BDT is set.

Figure 3-36 also demonstrates the pipelined protocol, this time with the initial timing field set to two, and the BDT field set to zero. In this example, $\overline{\text{LBA}}$ asserts at the beginning of the bus cycle, and negates at the end of the bus cycle so that a new address in loaded into the external memory on every rising edge of BCLK. The only difference between cycle #2 in **Figure 3-36** and cycle #1 in **Figure 3-37** is then number of wait states that are inserted before the first beat of data is transferred. Both cycle #2's are identical.

Figure 3-37 demonstrates the pipelined protocol, but with $\overline{\text{HPCE}}$ off at the start of the bus cycle. As in the burst protocol, when $\overline{\text{HPCE}}$ is initially off, the BCS will assert $\overline{\text{HPCE}}$, and will insert a wait state before the assertion of $\overline{\text{LBA}}$. Once $\overline{\text{LBA}}$ is asserted, the rest of the bus cycle will progress as in **Figure 3-38**.

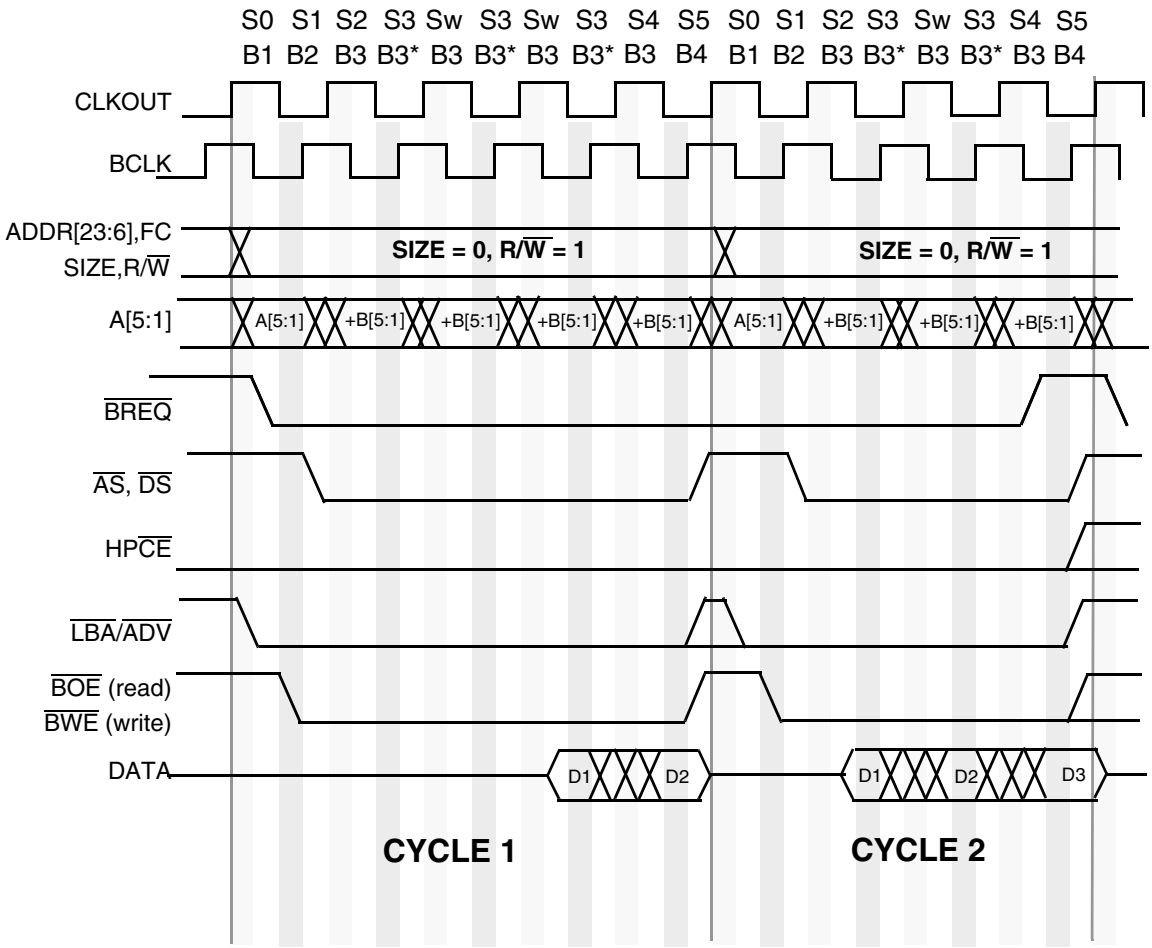
NOTE

When $\overline{\text{HPCE}}$ is off at the start of a bus cycle, the initial address will be valid for two clocks instead of just one.



Cycle 1: Initial Timing = 1, BDT = 1, HPCE is initially on.
Cycle 2: Initial Timing = 0, Burst Data Timing = 0

Figure 3-35 Pipelined Read Protocol (MT = 10)



Cycle 1: Initial Timing = 2, BDT = 0, HPCE is initially on.
Cycle 2: Initial Timing = 0, BDT = 0

Figure 3-36 Pipelined Read Protocol (MT = 10)

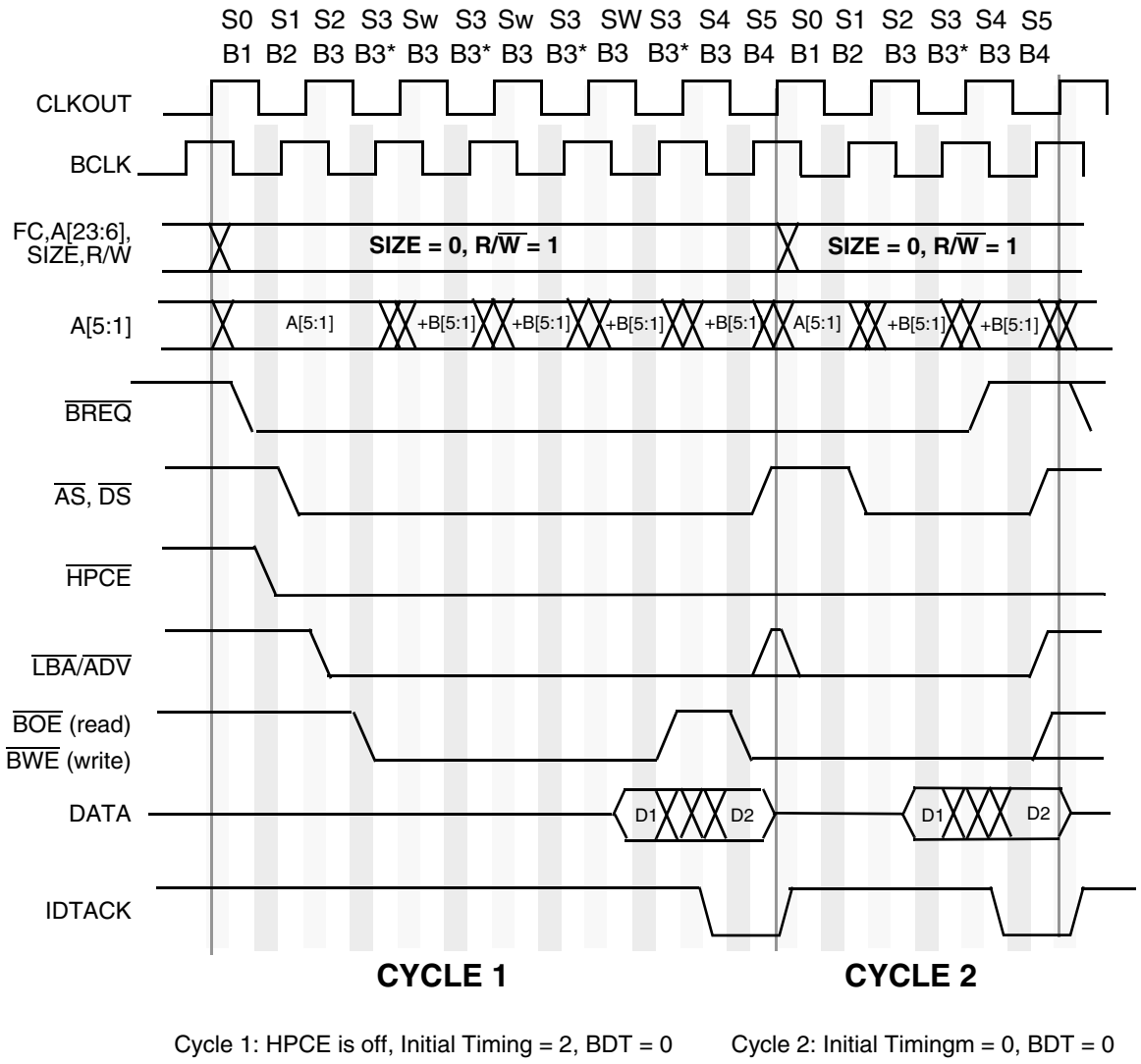
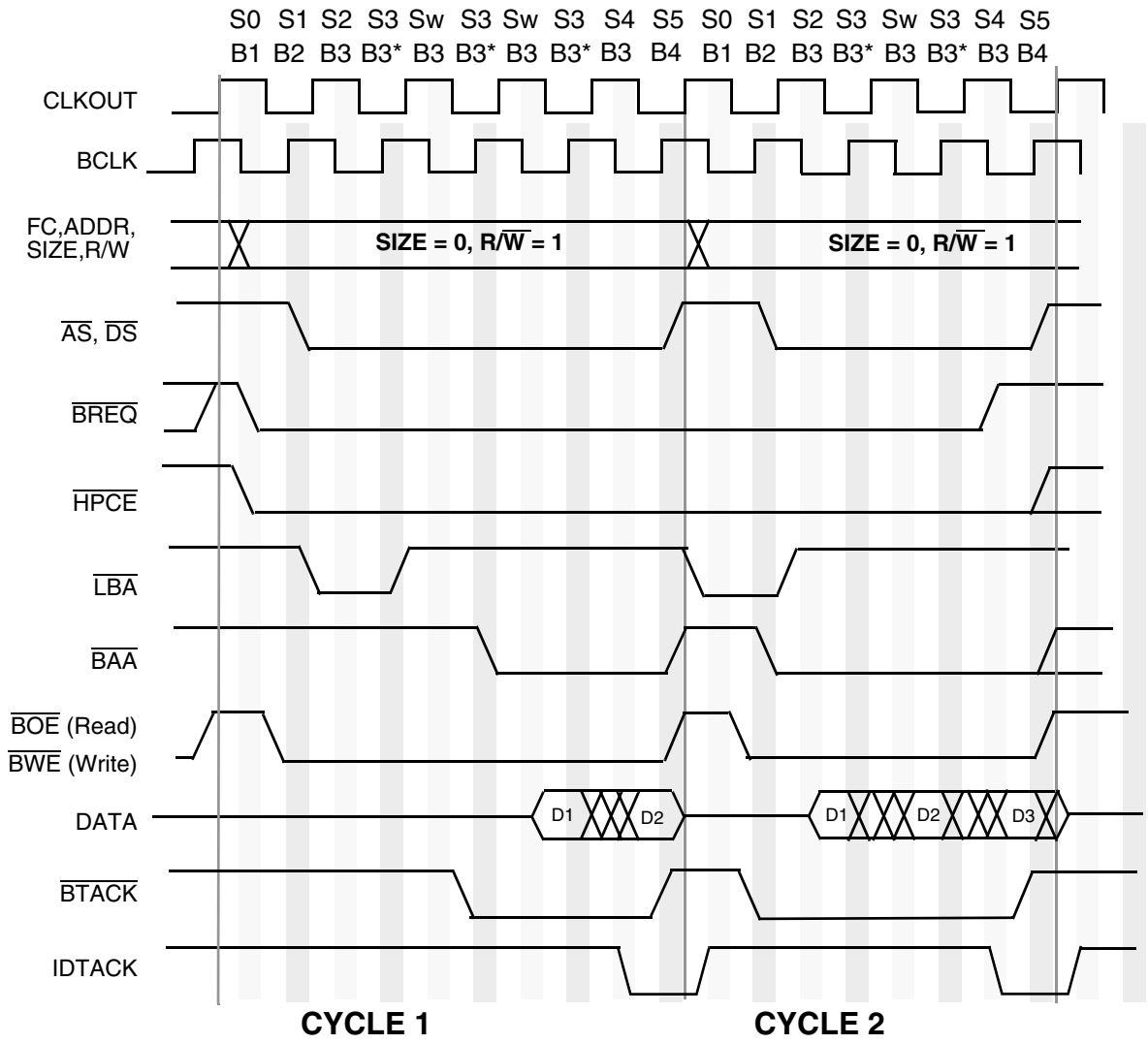


Figure 3-37 Pipelined Read Protocol (MT = 10)



Cycle 1: HPCE is initally off, Initial Timing = 1, BDT = 0; Cycle 2:Initial Timing = 0, BDT = 0

Figure 3-38 Burst Read Protocol, (MT = 00 or 01)

3.8 External Bus Interface (EBI)

The external bus interface (EBI) is the section of the BIM responsible for controlling the transfer of information between the IMB and external address space.

The EBI functions as a bus master or as a slave, depending on the BIM operation mode. In some BIM operation modes the EBI is not active. In master mode, emulation mode the EBI operates as the external bus master and allows internal bus cycles to access external resources. The EBI also gives limited support to alternate external masters with the external bus request ($\overline{EBR}$) signal. Refer to **3.8.7.1 External Bus Arbitration** for more details.

The EBI supports data transfers to both 16-bit and 8-bit devices. The chip select channels are programmed to define the device size for specific address ranges. When no chip select is active during an external data transfer, the device size is assumed to be 16 bits.

The EBI supports a variable length external bus cycle to accommodate the access speed of any device. During an external data transfer, the EBI drives the address, function codes, size and read/write information. $\overline{AS}$ signals the start of the external bus cycle, and $\overline{DS}$ signals valid data on a write cycle. Wait states are inserted until the bus cycle is terminated by the assertion of the internal DTACK signal by a chip select channel, or by the assertion of the external $\overline{DTACK}$ or $\overline{BERR}$ pins. For more details on chip select programming, refer to **3.6 Asynchronous Chip Select (ACS)**.

The minimum external bus cycle is two clocks. Termination of two clock bus cycles is hard to achieve because of $\overline{AS}$ timing. Typically, two clock bus cycles are terminated using a chip select to generate DTACK internally. The minimum 2 clock external bus cycle can also be achieved by asserting the $\overline{DTACK}$ pin or by asserting the $\overline{BERR}$ pin as specified by the AC timing requirements for two cycle termination. Generation of external $\overline{DTACK}$, $\overline{BTACK}$, or $\overline{BERR}$ based solely on addresses decoding without $\overline{AS}$ qualification will result in valid cycle termination since the termination signal(s) are not recognized by the EBI until the correct time in the external cycle (after addresses have been valid for at least one half clock).

An internal data transfer is terminated by the BIM or other IMB module. Internal cycles cannot be terminated by a chip select channel, external $\overline{DTACK}$ assertion, or external $\overline{BERR}$ assertion. Internal cycles may or may not have wait states.

In slave access mode (SLAM), the EBI operates as a slave interface to the external bus, allowing external bus cycles to access internal resources. Slave accesses are typically to internal memory locations. If a slave access is to an external address, the EBI does not present the bus cycle on the external bus. However, other BIM submodules, like the chip selects, respond normally. Alternate bus masters are not supported via $\overline{EBR}$ in this mode.

In single chip mode, the EBI is not active. All data transfers are between internal modules. $\overline{EBR}$ is meaningless in this mode because the external bus is never driven by the EBI, and show cycles are not supported.

Background debug mode is supported by the BIM in master mode, emulation mode, and single chip mode. The EBI operation is unaffected by background debug mode. Background debug mode operation in SLAM is not defined.

3.8.1 Security Mode

The BIM provides a security feature which disables all external reads of internal memory and registers, showcycle monitoring of internal bus activity, and visibility data bus activity. Security mode is enabled during reset by the state of a mask ROM bit. Security mode is enabled if the ROM bit is programmed as a 1. If enabled, security mode disables background debug mode, show cycles in master modes, external read and

write cycles in factory test modes and the visibility data bus. For testing purposes and authorized customer access, security mode can be disabled.

3.8.2 Bus Master Operation

3.8.3 Operand Transfer

From a CPU point of view, the possible operand accesses are: even byte, odd byte, even word, odd (misaligned) word, even long-word, and odd (misaligned) long-word. The even byte, odd byte, and even word accesses appear directly on the IMB. An even long-word access is presented to the IMB as two even word accesses.

The CPU32 does not support misaligned accesses and takes an exception. So, there are only four IMB access cases that must be handled by the BIM: even byte, odd byte, even word, and odd word. See **Table 3-89** for a listing of all supported transfer cases.

Table 3-89 MCU Interface to Various Port Sizes

Operand	OP0	OP1	OP2	OP3
31		OP0	OP1	OP2
	23		OP0	OP1
		15		OP0
			7	0
			Data Bus	

Case	Transfer Case	SIZ1	SIZ0	A0	DSACK1	$\overline{\text{DSACK0}}$	D15 D8	D7 D0
a	Byte to byte	0	1	X[1]	1	0	OP0	(OP0)[2]
b	Byte to word (even)	0	1	0	0	X	OP0	(OP0)
c	Byte to word (odd)	0	1	1	0	X	(OP0)	OP0
d	Word to byte (aligned)	1	0	0	1	0	OP0	(OP1)
e	Word to byte (misaligned)[3]	1	0	1	1	0	OP0	(OP0)
f	Word to word (aligned)	1	0	0	0	X	OP0	OP1
g	Word to word (misaligned)[3]	1	0	1	0	X	(OP0)	OP0
h	3-Byte to byte (aligned)[3,4]	1	1	0	1	0	OP0	(OP1)
i	3-Byte to byte (misaligned)[3,4]	1	1	1	1	0	OP0	(OP0)
j	3-Byte to word (aligned)[3,4]	1	1	0	0	X	OP0	OP1
k	3-Byte to word (misaligned)[3,4]	1	1	1	0	X	(OP0)	OP0
l	Long-word to byte (aligned)	0	0	0	1	0	OP0	(OP1)
m	Long-word to byte (misaligned)[3]	0	0	1	1	0	OP0	(OP0)
n	Long-word to word (aligned)	0	0	0	0	X	OP0	OP1
o	Long-word to word (misligned)[3]	0	0	1	0	X	(OP0)	OP0

NOTES:
1. X in a column means that the state of the signal has no effect.
2. Operands in parentheses are ignored by the CPU during read cycles.
3. Misaligned transfer cased not supported.
4. Three-byte transfer cases occur only as a result of an aligned long-word to byte.

The EBI controls the byte or word operand transfers between the IMB and 8-bit or 16-bit devices. Each device is assigned to particular bits of the data bus. A 16-bit device is assigned to data bus bits [15:0] and an 8-bit device is assigned to data bus bits [15:8]. Depending on the device size, the EBI may run more than one external bus cycle to complete the operand transfer. Signals issued by the EBI control data movement on the external bus.

In the operand transfer cases that follow, the EBI is the exernal bus master. The upper portion of each figure shows the operand for the current cycle. Its position indicates the location of the data on the IMB data bus. OP0 is the most significant byte, if the operand contains two bytes of data. For each external bus cycle performed, “OPx” indicates where operand byte “x” is driven on the 16-bit external data bus.

3.8.3.1 Byte Operand, 8-Bit Device, Even (A0 = 0)

For a byte transfer between an even memory location and an 8-bit device, the EBI drives the external address bus with the desired address and drives the SIZE pin high to indicate a byte operand transfer. In the case of a read, the slave device responds by placing the data on bits [15:8] of the data bus. In the case of a write, the EBI drives the single byte operand on bits [15:8] of the external data bus.

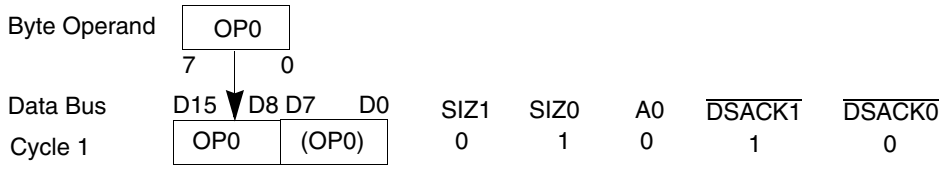


Figure 3-39 Even Byte Transfer to 8-Bit Device

3.8.3.2 Byte Operand, 8-Bit Device, Odd (A0 = 1)

For a byte transfer between an odd memory location and an 8-bit device, the EBI drives the external address bus with the desired address and drives the SIZE pin high to indicate a byte operand transfer. In the case of a read, the slave device responds by placing the data on bits [15:8] of the data bus. In the case of a write, the EBI drives the single byte operand on bits [15:8] of the external data bus.

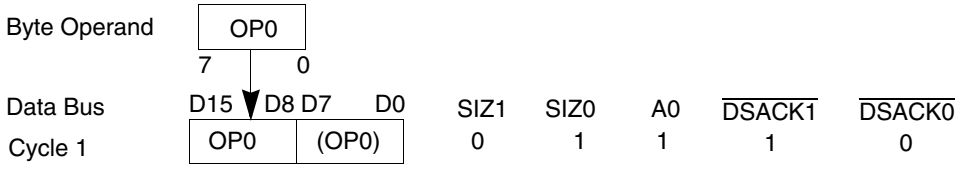


Figure 3-40 Byte Operand to 8-Bit Device, Odd (A0 = 1)

3.8.3.3 Byte Operand, 16-Bit Device, Even (A0 = 0)

For a byte transfer between an even memory location and a 16-bit device, the EBI drives the address bus with the desired address and drives the SIZE pin high to indicate a byte operand transfer. In the case of a read, the slave device places the data on bits [15:8] of the data bus. The EBI ignores bits [7:0]. In the case of a write, the EBI drives the single byte operand on bits [15:8] of the data bus. Bits [7:0] are not driven.

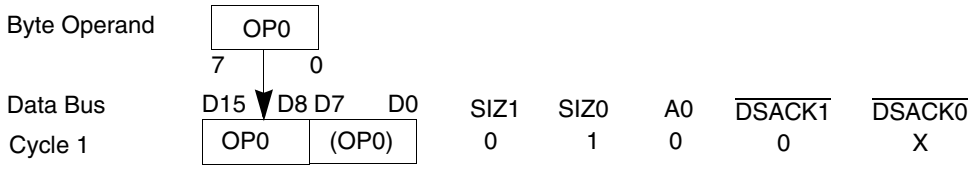


Figure 3-41 Byte Operand to 16-Bit Device, Even (A0 = 0)

3.8.3.4 Byte Operand, 16-Bit Device, Odd (A0 = 1)

For a byte transfer between an odd memory location and a 16-bit device, the EBI drives the address bus with the desired address and drives the SIZE pin high to indicate a byte operand transfer. In the case of a read the slave device places the data on bits [7:0] of the data bus. The EBI ignores bits [15:8]. In the case of a write, the EBI drives the single byte operand on bits [7:0] of the data bus. Bits [15:8] are not driven.

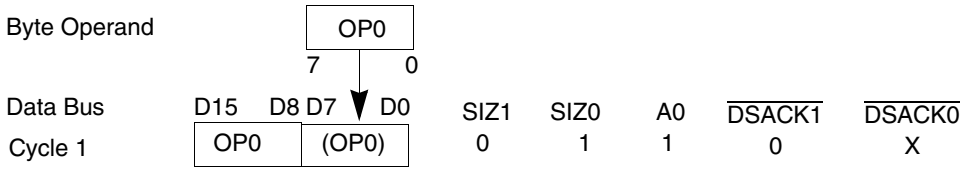


Figure 3-42 Byte Operand to 16-Bit Device, Odd (A0 = 1)

3.8.3.5 Word Operand, 8-Bit Device, Even (A0 = 0)

For a word transfer between an even memory location and an 8-bit device, two external bus cycles are performed. First, the EBI drives the address bus with the even address and drives the SIZE pin low to indicate a word operand transfer. In the case of a read, the slave device places the data on bits [15:8] of the data bus. In the case of a write, the EBI drives the most significant byte of the operand on bits [15:8] of the data bus. In the second bus cycle the EBI drives the address bus with the odd address

and the SIZE pin high to indicate a byte operand transfer. In the case of a read, the slave device places the data on bits [15:8] of the data bus. In the case of a write, the EBI drives the least significant byte of the operand on bits [15:8] of the data bus.

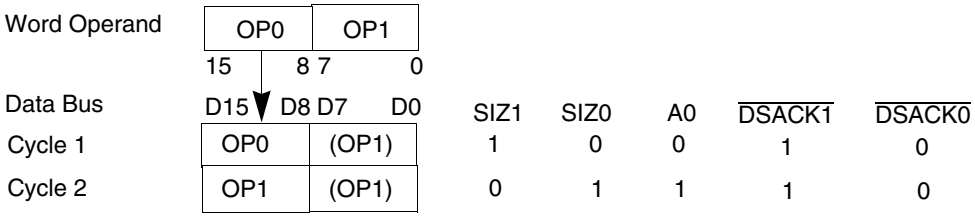


Figure 3-43 Word Operand to 8-Bit Device, Aligned

3.8.3.6 Word Operand, 16-Bit Device, Even (A0 = 0)

For a word transfer between an even memory location and a 16-bit device, the EBI drives the address bus with the desired address, and drives the SIZE pin low to indicate a word operand transfer. In the case of a read, the slave device responds by placing the data on bits [15:0] of the data bus. In the case of a write, the EBI drives the word operand on bits [15:0] of the data bus.

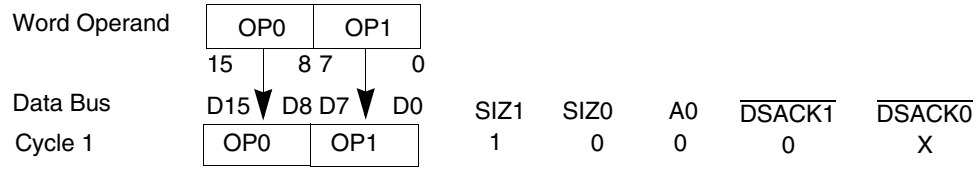


Figure 3-44 Word Operand to 16-Bit Device, Aligned

3.8.3.7 Long-Word Operand to 16-Bit Device, Aligned

Figure 3-45 shows both a long-word and word read and write timing to a 16-bit device. The MCU drives the address bus with the desired address and drives the size pins to indicate a long-word operand.

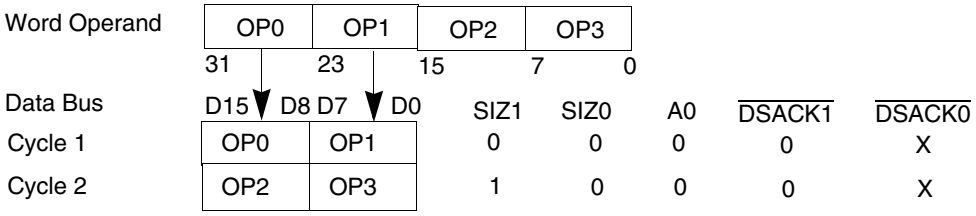


Figure 3-45 Long-Word Operand to 16-Bit Device, Aligned

For a read operation, the slave responds by placing the two most significant bytes of the operand on bits [15:0] of the data bus, and asserting DSACK1 to indicate a 16-bit device. The MCU reads the two most significant bytes of the size counter, increments the address, initiates a new cycle, and reads bytes 2 and 3 of the operand from bits [15:0] of the data bus.

For a write operation, the MCU drives the two most significant bytes of the operand on bits [15:0] of the data bus. The slave device then reads the two most significant bytes of the operand (bytes 0 and 1) from bits [15:0] of the data bus and asserts DSACK1 to indicate that it received the data and is a 16-bit device. The MCU then decrements the transfer size counter by two, increments the address by two, and writes bytes 2 and 3 of the operand to bits [15:0] of the data bus.

3.8.3.8 Long-Word Operand to 8-Bit Device, Aligned

The MCU drives the address bus with the desired address and the size pins to indicate a long-word operand, see **Figure 3-46**.

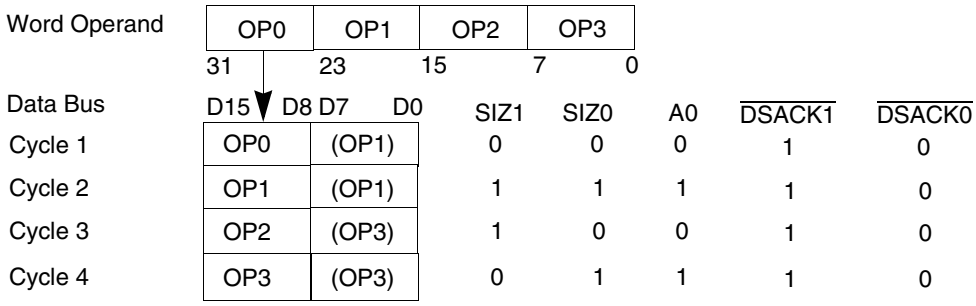


Figure 3-46 Long-Word Operand to 8-Bit Device, Aligned

For a read operation, the slave places the most significant byte of the operand on bits [15:8] of the data bus, and asserting DSACK0 to indicate an 8-bit device. The MCU

reads the most significant byte of the operand (byte 0) from bits [15:8] and ignores bits [7:0]. The MCU then decrements the transfer size counter, increments the address, initiates a new cycle, and reads byte 1 of the operand from bits [15:8] of the data bus. The MCU repeats the process of decrementing the transfer size counter, incrementing the address, initiating a new cycle and reading a byte to trnasfer the remaining two bytes.

For a write operation, the MCU drives the two most significant bytes of the operand on bits [15:0] of the data bus. The slave divice then reads only the most significant byte of the operand (byte 0) from bits [15:8] of the data bus and asserts $\overline{\text{DSACK0}}$ (but not $\overline{\text{DSACK1}}$) to indicate that it received the data and it is an 8-bit device. The MCU then decrements the transfer size counter, increments the address, and writes byte 1 of the operand to bits [15:8] of the data bus. The MCU continues to decrement the transfer size counter, increment the address, and write a byte to transfer the remaining two bytes to the slave device.

3.8.4 Bus Master Cycles

In this section, each EBI bus cycle type is defined in terms of actions associated with a succession of internal states. These internal states are only for reference.

Read or write operations may require multiple bus cycles to complete based on the operand size, operand alignment, and target device size. Refer to **3.8.3 Operand Transfer** for more information. In the discussion that follows, it is assumed that only a single bus cycle is required to transfer the data.

In the timing diagrams, data transfers are related to clock cycles, independent of the clock frequency. The external bus states are also referenced.

Note that WAIT states on the BIM EBI are counted differently than on previous Integration Modules. A BIM 0 wait state cycle is a 2-clock bus cycle.

3.8.4.1 Read Cycles

During a read cycle, the EBI receives data from a memory or peripheral device. A flowchart of a typical read cycle operation is shown in **Figure 3-47**. On **Figure 3-49** and **Figure 3-50** are timing diagrams of external bus master cycles with and without wait states. The timing diagrams also show IMB bus activity to assist in understanding. The EBI does support delayed ds cycles, although this case is not shown.

- State 0 — The EBI drives the address bus and function codes. R/$\overline{\text{W}}$ is driven high, indicating a read cycle, and the SIZE pin is driven to indicate the number of bytes in the transfer.
- State 1 — One-half clock later in S1, the EBI asserts $\overline{\text{AS}}$, indicating that the address on the address bus is valid. $\overline{\text{DS}}$ is asserted indicating that the EBI is ready for data. $\overline{\text{CS}}$[1] may be asserted with either $\overline{\text{AS}}$ or $\overline{\text{DS}}$.
  The selected device uses R/$\overline{\text{W}}$, SIZE, A[0], and $\overline{\text{DS}}$ to place its information on the data bus (D[15:8] and/or D[7:0]). One or both bytes of the data bus are selected by SIZE and A[0].

1. A chip select pin may not be available in all BIM configurations.

- State 2 — No new control signals are issued during S2.
  If $\overline{\text{DTACK}}$[2] was asserted before the beginning of S2, the EBI proceeds to State 5.
- Optional Wait States — Wait states[3] are inserted until the slave asserts $\overline{\text{DTACK}}$.
- State 5 — During S5, the incoming data is latched into the internal data bus holding register. $\overline{\text{CS}}$, $\overline{\text{AS}}$ and $\overline{\text{DS}}$ are negated. The address bus, function codes, R/$\overline{\text{W}}$, and SIZE remain valid through S5 to allow for static memory operation and signal skew.
  The slave device asserts $\overline{\text{DTACK}}$ and data until it detects the negation of $\overline{\text{AS}}$ or $\overline{\text{DS}}$ (whichever is detected first). The slave must remove its data and negate $\overline{\text{DTACK}}$ within approximately one-half of a clock period after recognizing the negation of $\overline{\text{AS}}$ or $\overline{\text{DS}}$. Note that the data bus is not free until S1.

2. $\overline{\text{DTACK}}$ can be asserted internally by a chip select channel or the $\overline{\text{DTACK}}$ pin can be asserted.
3. Wait states are counted in full clocks, two half-clock tics each.

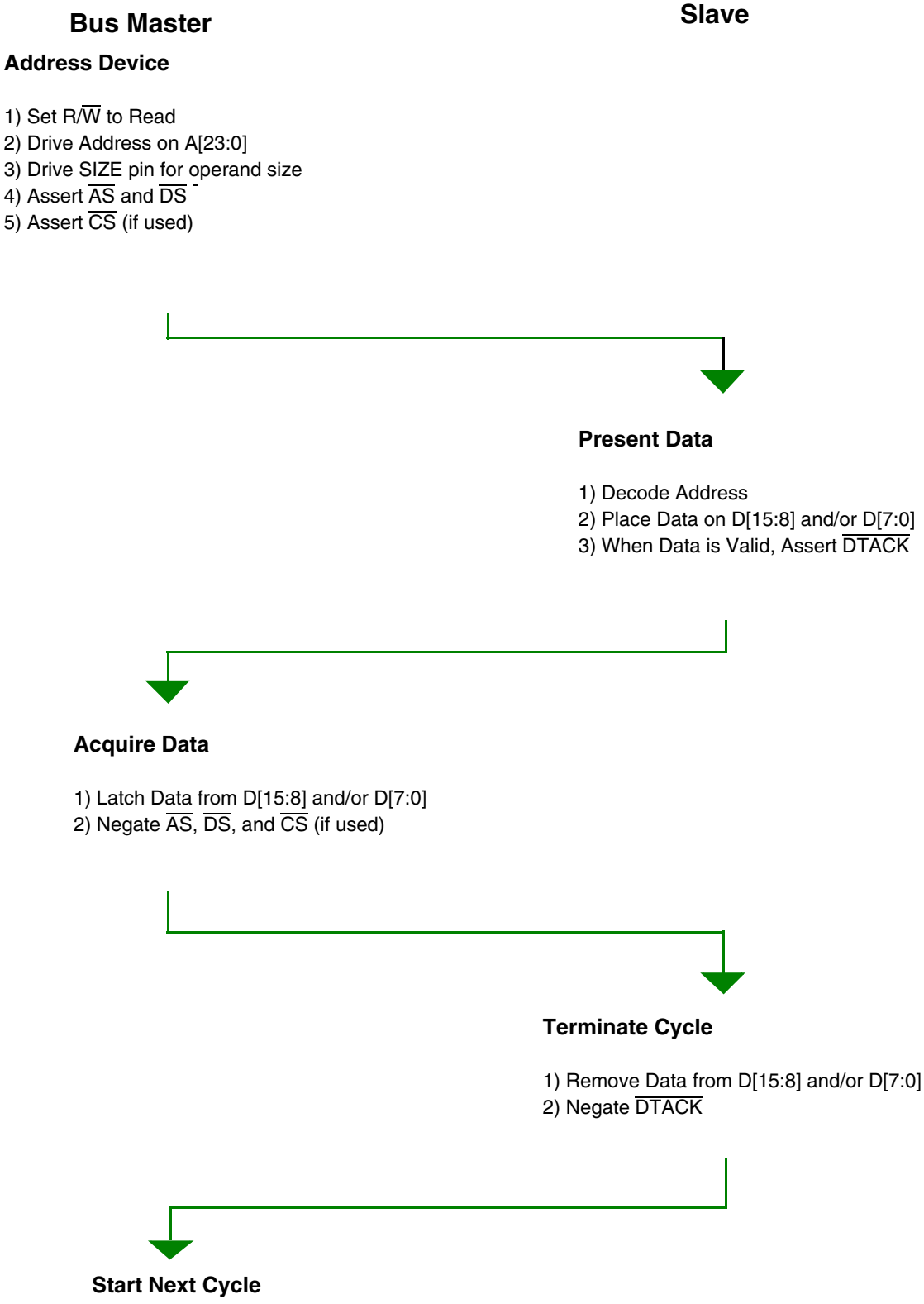


Figure 3-47 Read Cycle Flowchart

3.8.4.2 Write Cycles

On a write cycle, the EBI transfers data to a memory or peripheral device. **Figure 3-48** is a flowchart of the write cycle operation. The timing diagrams also show IMB bus activity to assist in understanding. The EBI does support delayed ds cycles, although this case is not shown.

- State 0 — The EBI drives the address bus and function codes. R/$\overline{W}$ is driven low, indicating a write cycle, and the SIZE pin is driven to indicate the number of bytes in the transfer.
- State 1 — One-half clock later in S1, the EBI asserts $\overline{AS}$, indicating that the address on the address bus is valid. $\overline{CS}$ may be asserted with $\overline{AS}$.
- State 2 — The EBI drives its data onto the data bus (D[15:0] and/or D[7:0]). The selected device uses R/$\overline{W}$, SIZE, and A[0] to take its information from the data bus. One or both bytes of the data bus are selected by SIZE and A[0].

  If $\overline{DTACK}$[4] is asserted by the slave device before the beginning of S2, the EBI proceeds to State 5.
- Optional Wait States — Wait states[5] are inserted until the slave asserts $\overline{DTACK}$. When data is stable on the data bus, $\overline{DS}$ is asserted. $\overline{CS}$ may be asserted with $\overline{DS}$.
- State 5 — During S5, $\overline{CS}$, $\overline{AS}$ and $\overline{DS}$ are negated. The address, function codes, R/$\overline{W}$, and SIZE remain valid through S5 to allow for static memory operation and signal skew.
  The EBI asserts data through S5.

  The slave must negate $\overline{DTACK}$ within approximately one-half of a clock period after recognizing the negation of $\overline{AS}$ or $\overline{DS}$.

4. $\overline{DTACK}$ can be asserted internally by a chip select channel or the $\overline{DTACK}$ pin can be asserted.
5. Wait states are counted in full clocks, two half-clock tics each.

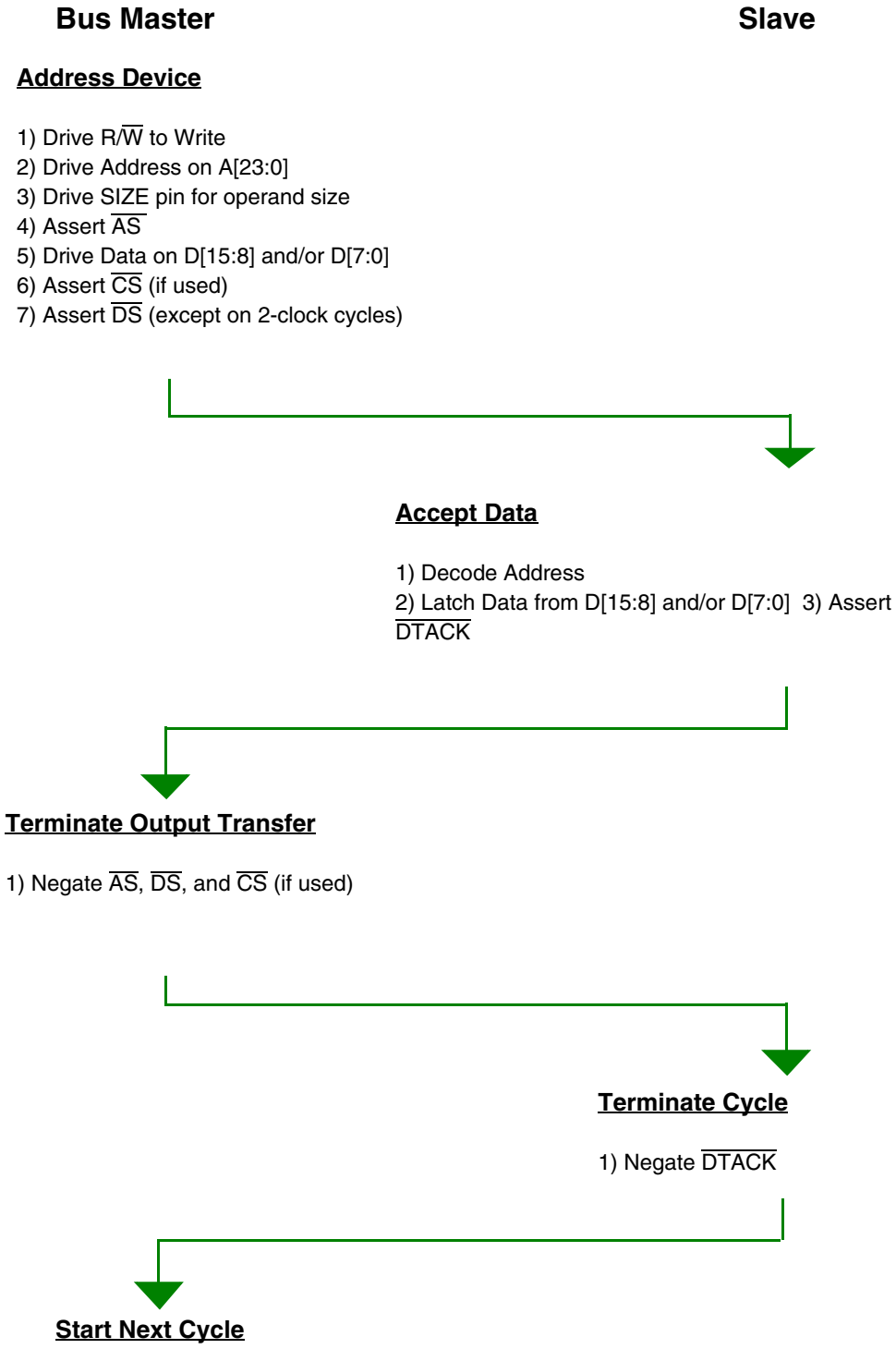


Figure 3-48 Write Cycle Flowchart

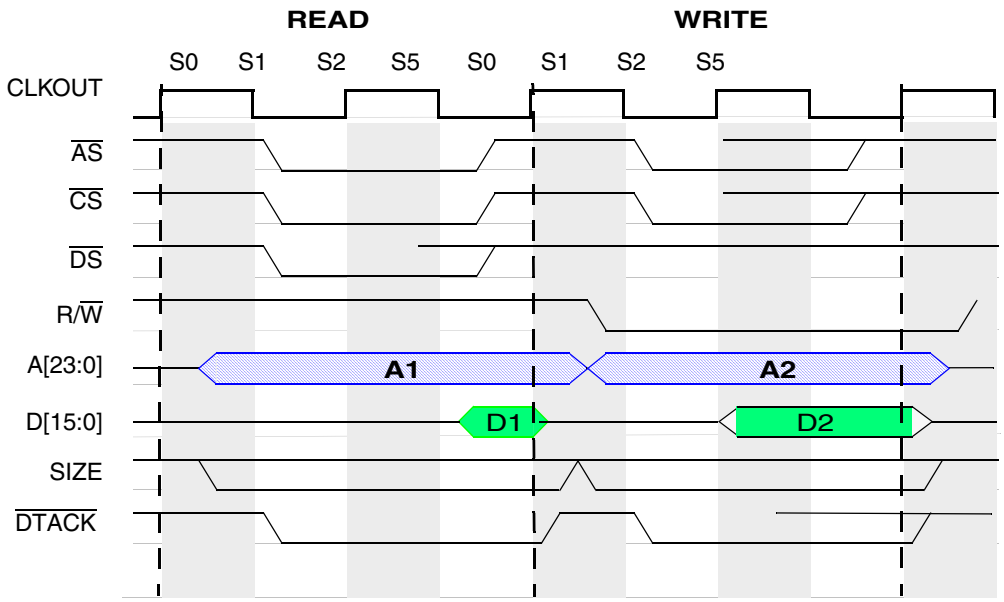


Figure 3-49 Master Mode — 2-Clock Bus Cycle

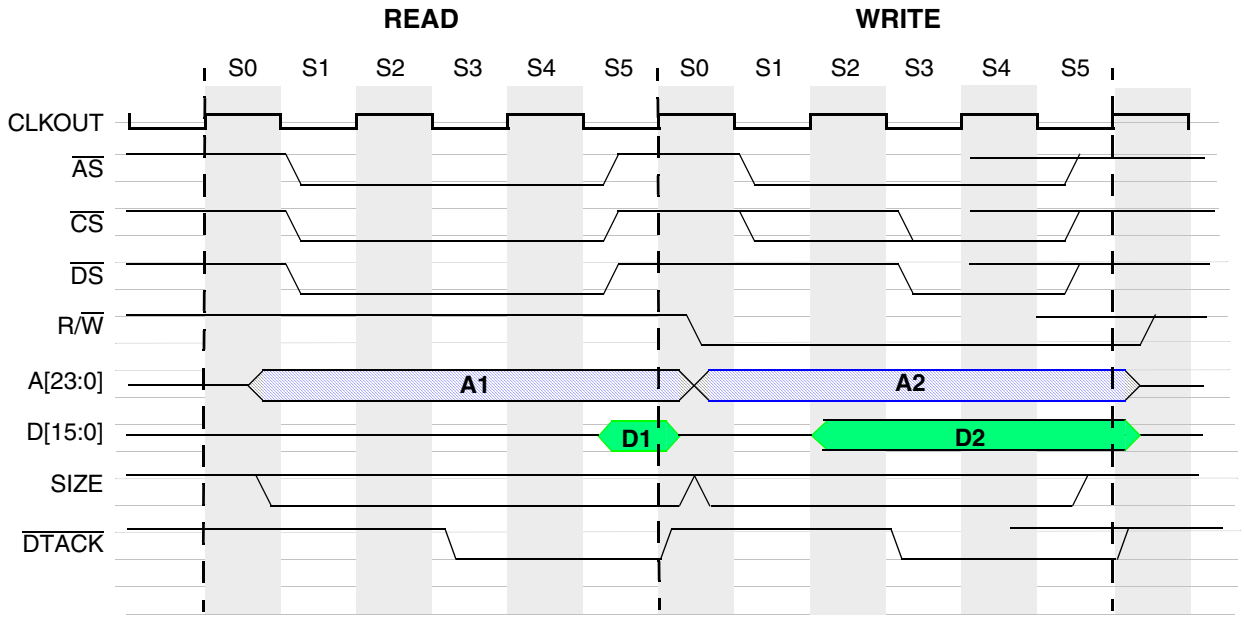


Figure 3-50 Master Mode — 3 Clock Bus Cycle

3.8.5 Burst Read and Write Cycles

The BIM is able to run synchronous burst cycles when a burst request is issued by a burst-capable CPU. Burst protocol, a superset of the basic cycle, allows one or more data transfers to be completed in a single bus cycle by stretching the DATA portion of the basic cycle. In a burst cycle the EBI issues the address of the first data item to be

transferred. The slave acknowledges data transfers by asserting $\overline{\text{BTACK}}$. Data transfers continue until either the master or the slave terminate the cycle by asserting $\overline{\text{DTACK}}$. The final transfer of a burst cycle occurs in state S5.

$\overline{\text{BTACK}}$ is both an indication that a device supports burst protocol and that the device is ready to transfer burst data. In most systems, $\overline{\text{BTACK}}$ cannot be tied to VSS, rather it must be asserted only for the burst address range decoded by the burst device. If $\overline{\text{BTACK}}$ is tied to V_{SS}, every external burst request must have a corresponding burst device capable of sustaining zero wait state burst access through it's entire address range. Finally, if $\overline{\text{BTACK}}$ is tied to VSS, an access to an unimplemented address will not cause the bus time-out monitor to assert internal $\overline{\text{BERR}}$.

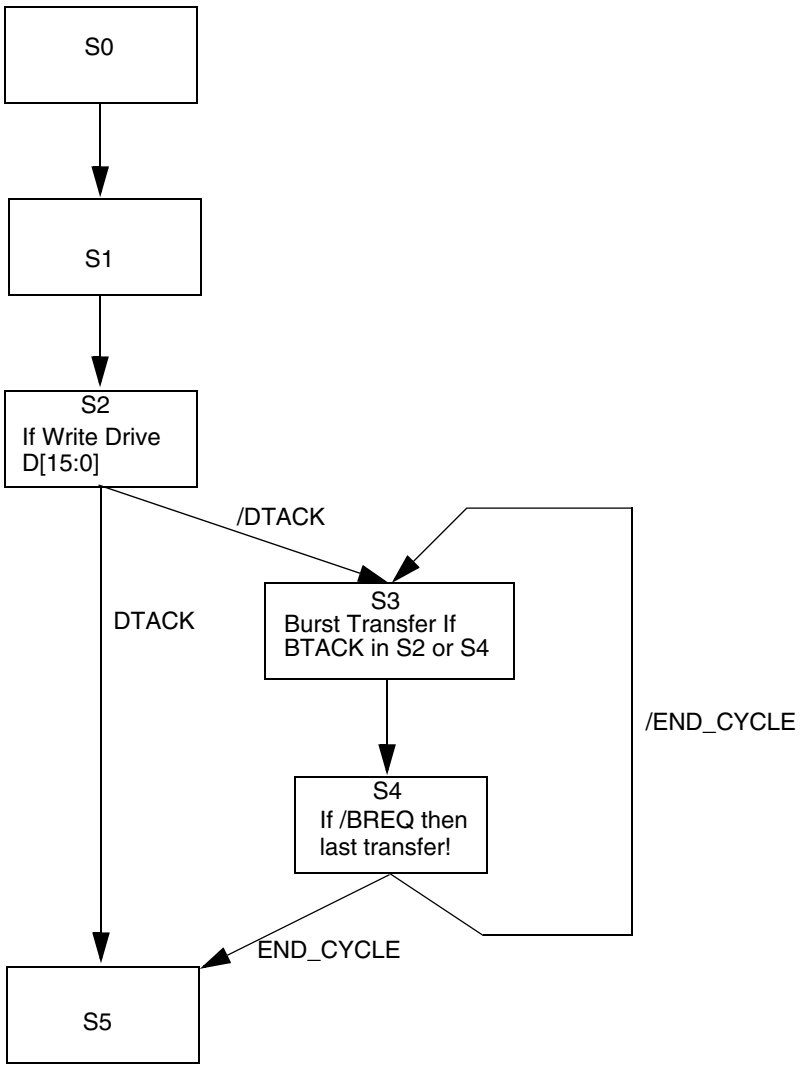
Both the master and the slave maintain the address of the data being transferred and increment the address each time data is successfully transferred. Every burst device is required to implement the same address modification algorithm. Burst cycles always transfer bus width aligned data; no partial bus width transfers, (i.e., 8-bit devices, are supported).

To support low latency bus operation required in microcontroller systems, the bus master may terminate a burst cycle by negating $\overline{\text{BREQ}}$, or a slave may terminate the burst cycle by asserting $\overline{\text{DTACK}}$ or $\overline{\text{BERR}}$. Due to burst timing handshake constraints, if the bus master terminates the burst cycle by negating $\overline{\text{BREQ}}$, the EBI will terminate the bus cycle for the burst device by internally asserting $\overline{\text{DTACK}}$ when the burst device indicates that it is ready to transfer data ($\overline{\text{BTACK}}$ asserts). If a reset is pending, the EBI also terminates burst read cycles by asserting $\overline{\text{DTACK}}$. Burst write cycles are allowed to complete when a reset is pending without EBI interference.

For increased cycle issue flexibility the bus master may convert a burst cycle to a basic cycle without a performance penalty by negating $\overline{\text{BREQ}}$ before the slave transfers the first data item (i.e. a burst length of 1 gracefully degrades to a two clock cycle). Additional waveform specifications for burst transfers can be found later in this section.

NOTE

At this time, the only burst-capable CPU is the CPU32X. In addition, the CPU32X only supports burst reads.



END_CYCLE = S3 * DTACK - slave terminates burst on current data transfer
+ S3 * /BREQ * BTACK - master terminates burst at next data transfer

Figure 3-51 Burst Cycle State Machine

3.8.5.1 Burst Read Cycles

During a burst read cycle, the EBI receives data from a burst memory device.

- State 0 —The EBI drives the address bus and function codes. $\overline{\text{BREQ}}$ is asserted, indicating the bus master is requesting a burst transfer. R/$\overline{\text{W}}$ is driven high, indi-

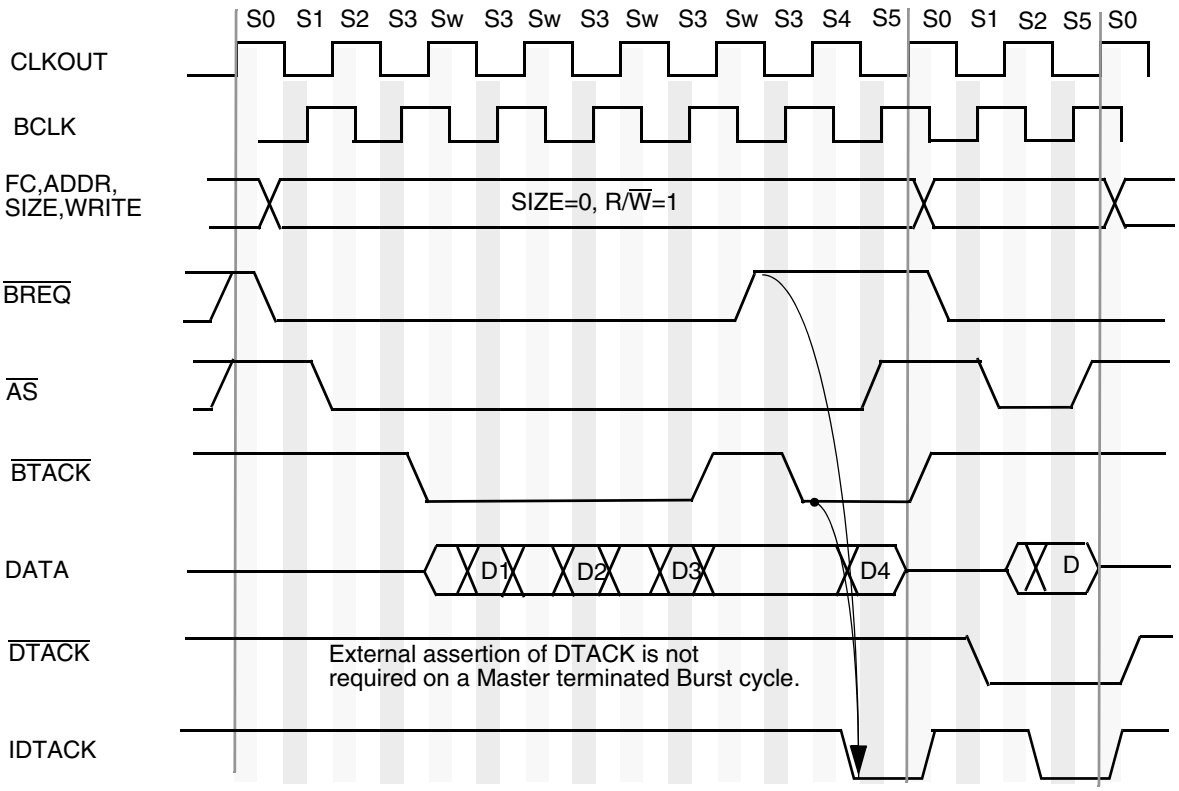
cating a read cycle, and the SIZE pin is driven low to indicate a word-size transfer.

- State 1 — The EBI asserts AS, DS, (LBA and BOE) indicating that the address on the address bus is valid and that the slave device may drive data on D[15:0]. The slave device may assert DTACK[6] by the end of S1 to convert the burst cycle to a 2 clock non-burst cycle (State sequence S0, S1, S2, S5).
- State 2 — If BTACK, burst data is to be transferred in S3. The EBI may negate BREQ in S2 to convert the burst cycle to a standard read cycle. LBA is negated. If DTACK then S5 else S3.
- State 3 — If BTACK was asserted previously, then at the beginning of S3 the burst device places data on D[15:0]. Midway through S3, the EBI latches data into the internal data bus holding register.
  If DTACK and BTACK are negated then a wait state is inserted by proceeding to SW.

  If DTACK is asserted the cycle will terminate, proceed to State 4.
- State SW — State SW is a wait state. EBI may negate BREQ in SW to terminate the burst cycle after the current data transfer has completed Proceed to State S3.
- State 4 — No signal changes.
- State 5 — At the beginning of S5 the burst device places the final data on D[15:0]. Midway through S5, the EBI latches data into the internal data bus holding register. AS, DS, and BOE are negated. The address bus, function codes, R/W, BREQ, and SIZE remain valid through S5 to allow for static memory operation and signal skew.
  The slave device negates DTACK, BTACK and its data before the end of S0.

The following burst read waveforms illustrate various burst cycle scenarios.

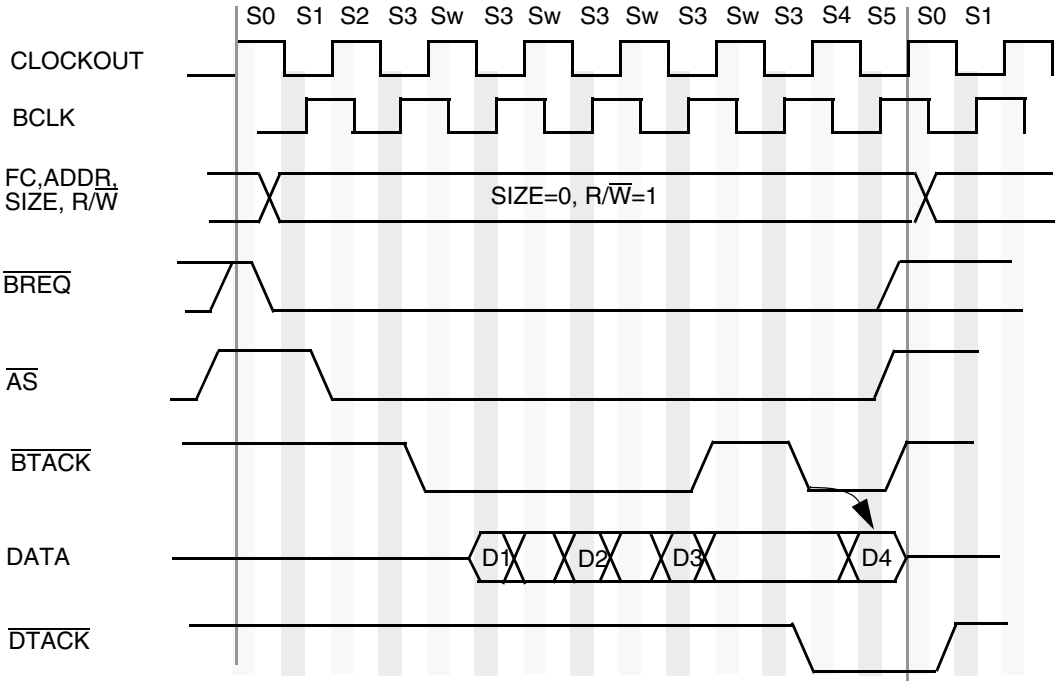
6. DTACK can be asserted internally by the Burst Chip Select or the DTACK pin can be asserted.



NOTE: BIM asserts IDTACK @ BTACK if Master terminates a burst cycle by negating BREQ.

Cycle 1: Master terminates burst cycle by negating BREQ. Slave inserts 1 wait on last transfer.
Cycle 2: Burst cycle terminated by slave after 1 data transfer without wait states.

Figure 3-52 Burst Read — Master Termination



Slave terminates burst cycle by asserting DTACK, inserts 1 Wait state in D4.

Figure 3-53 Burst Read — Slave Termination

3.8.5.2 Read-Modify-Write (RMC)

When the read-modify-write instruction is executed, an internal, indivisible read-modify-write cycle is performed. The CPU first performs a read, modifies the data, and then writes the data back to the same address. The BIM does NOT provide external indication that a read-modify-write cycle is in progress.

For both the read and write cycles, multiple bus cycles may be required to transfer the entire operand. The external bus may be arbitrated away from the BIM at any point during the read-modify-write operation.

3.8.6 CPU Space Cycles

CPU space cycles are used for LPSTOP broadcast cycles, interrupt acknowledge cycles, and breakpoint acknowledge cycles. The CPU space type is encoded on bits [19:16] of the address bus and the function codes are driven to 0x7 to distinguish these cycles. ***Refer to the User's Manual for the appropriate CPU for a complete description of CPU space cycle encoding.***

CPU space cycles are handled like other data transfers. The EBI drives the address, function codes, SIZE and R/W signals. $\overline{AS}$ remains negated for all CPU space cycles, internal or external. For applications that require an external indication of CPU space cycles, a chip select may be programmed to assert. See **3.6.4 Asynchronous Chip Select Operations**.

3.8.6.1 LPSTOP Broadcast Cycles

The LPSTOP write cycle is generated by the CPU when executing the LPSTOP instruction. The BIM latches the interrupt mask value driven on the data bus by the CPU and terminates the cycle internally. The BIM does not drive the interrupt mask value on the external data bus, unless show cycles is enabled. A chip select may be programmed to assert on LPSTOP broadcast cycles. Refer to **3.1.8 LPSTOP Operation** for an LPSTOP overview.

While in LPSTOP mode, the $\overline{\text{EBR}}$ pin controls whether the external pins are driven. If $\overline{\text{EBR}}$ is asserted, the BIM does not drive the address, data, $\overline{\text{AS}}$, $\overline{\text{DS}}$, R/$\overline{\text{W}}$, $\overline{\text{SIZE}}$, and $\overline{\text{BERR}}$ pins. If $\overline{\text{EBR}}$ is negated, the BIM drives the pins to the state they were in when LPSTOP was entered. If the state of $\overline{\text{EBR}}$ is changed during LPSTOP, the EBI reacts accordingly.

3.8.6.2 Breakpoint Acknowledge Cycles

The CPU performs a breakpoint acknowledge cycle in response to a BKPT instruction or the assertion of the $\overline{\text{BKPT}}$ pin. When the breakpoint acknowledge cycle is terminated internally by an IMB module and show cycles is enabled, the EBI drives the external data bus to reflect the internal data activity. When the breakpoint acknowledge cycle is an external cycle, $\overline{\text{AS}}$ is not asserted. A chip select pin can be programmed to assert for external breakpoint cycles.

3.8.6.3 Interrupt Acknowledge Cycles

An interrupt acknowledge cycle is performed to service an interrupt request. When the interrupt acknowledge cycle is terminated internally by an IMB module the EBI drives the external data bus to reflect the internal data bus activity. When the interrupt acknowledge cycle is to service an interrupt request from an IRQ pin and an autovector is not selected, the EBI performs an external interrupt acknowledge cycle to acquire the interrupt vector.

An external interrupt acknowledge flow chart is shown in **Figure 3-54**, and a timing diagram for external IACK cycles is shown in **Figure 3-55**. Refer to **3.1.7 Interrupt Operation** for an interrupt overview.

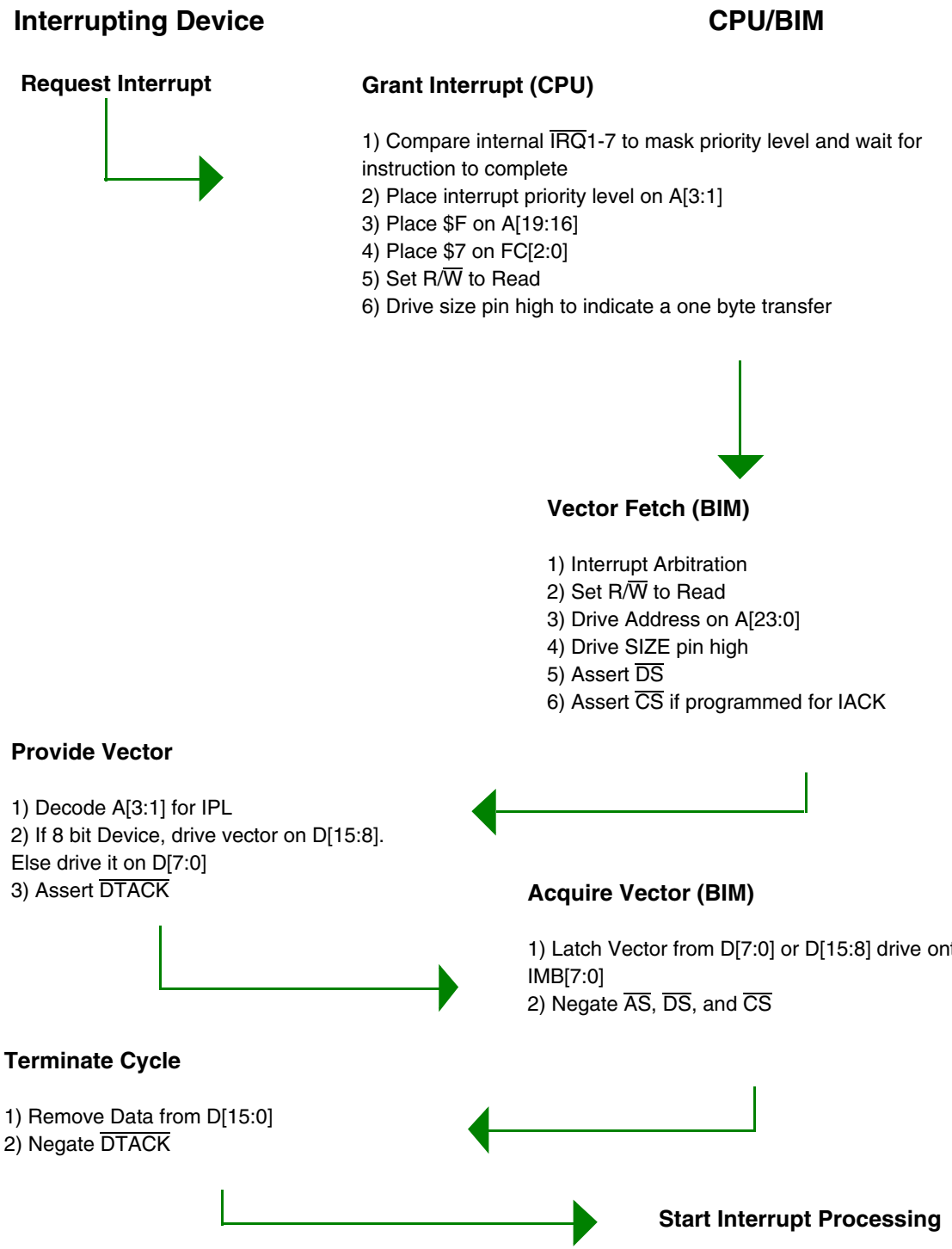


Figure 3-54 IACK Sequence Flow, External Vector

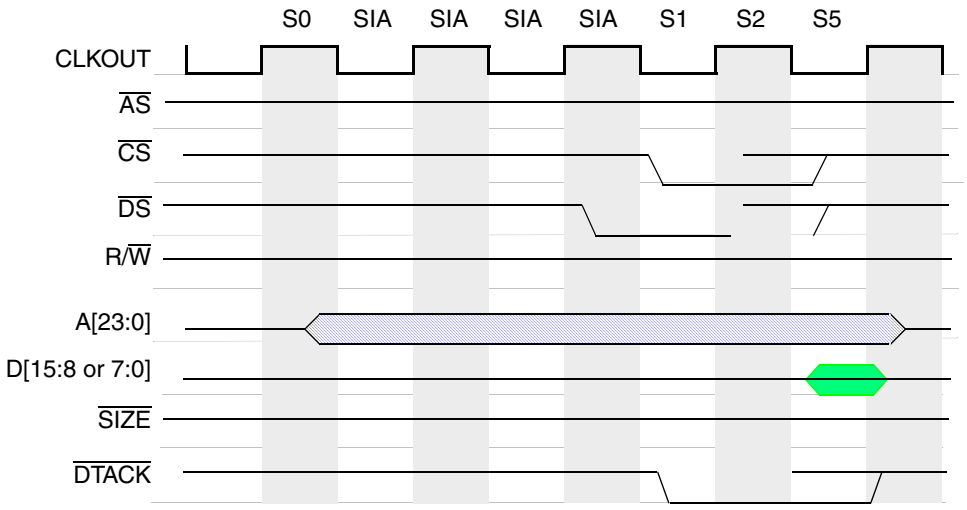


Figure 3-55 IACK Cycle — External Vector

3.8.7 Bus Exception Operation

Normal bus cycle termination requires the assertion of the DTACK pin or the internal IDTACK signal. Minimal bus exception support is provided by bus error cycle termination. For bus error cycle termination, the external BERR pin or the internal IBERR signal is asserted. Bus error cycle termination takes precedence over normal cycle termination, provided BERR assertion meets the timing constraints described in **E.2.2 BIM AC Timing**.

The BIM provides an internal bus monitor which optionally asserts the internal IBERR signal when DTACK or BTACK response time is too long. Also, a spurious interrupt monitor terminates IACK cycles asserting the internal IBERR signal, when no interrupt arbitration occurs. For more information on these internal IBERR functions, see **3.5 System Protection**.

Address error execptions are also supported by the BIM. Other **MC68000** family bus exceptions, (late bus error, retry, relinquish and retry, and halt), are **not** supported by the BIM.

3.8.7.1 External Bus Arbitration

The BIM external bus has limited support for an alternate bus master via the external bus request (EBR) signal. When EBR is asserted, the BIM completes its bus cycle, and relinquishes control of the external bus. After AS and DS have been negated for three clock cycles, the alternate master may take control of the external bus. The alternate master cannot access any MCU internal resources.

If the external bus is granted to an alternate bus master in master, MFTM, emulation, or background debug operation mode, the pins that tri-state depend on the pin function specified in the pin assignment register. Pins programmed to function as A[23:0],

FC[2:0], D[15:0], $\overline{\text{BREQ}}$, $\overline{\text{AS}}$, $\overline{\text{DS}}$, R/$\overline{\text{W}}$, SIZE, $\overline{\text{CS1}}$ – $\overline{\text{CS7}}$, $\overline{\text{LBA}}$, $\overline{\text{BAA}}$, $\overline{\text{BWE}}$, and $\overline{\text{BOE}}$ tristate.

$\overline{\text{EBR}}$ operation is not defined in SLAM or single chip mode. Asserting $\overline{\text{EBR}}$ causes undefined pin operation.

While $\overline{\text{EBR}}$ is asserted, the CPU will continue to execute internal bus cycles until an external bus cycle is needed. At this point, the BIM will disable its bus time-out logic and insert wait states on the IMB until the external bus master has relinquished the bus mastership. Pins programmed to function as DTACK, BTACK, BERR, and BKPT have no affect on the internal bus cycle execution when the external bus is granted to an alternate master.

Because the external bus may be relinquished at bus cycle boundaries, data coherency for external long-word or misaligned accesses is not maintained. If show cycles are enabled, data coherency for read-modify-write cycles and internal longword or misaligned accesses is not maintained. Refer to **Figure 3-56** for a timing diagram.

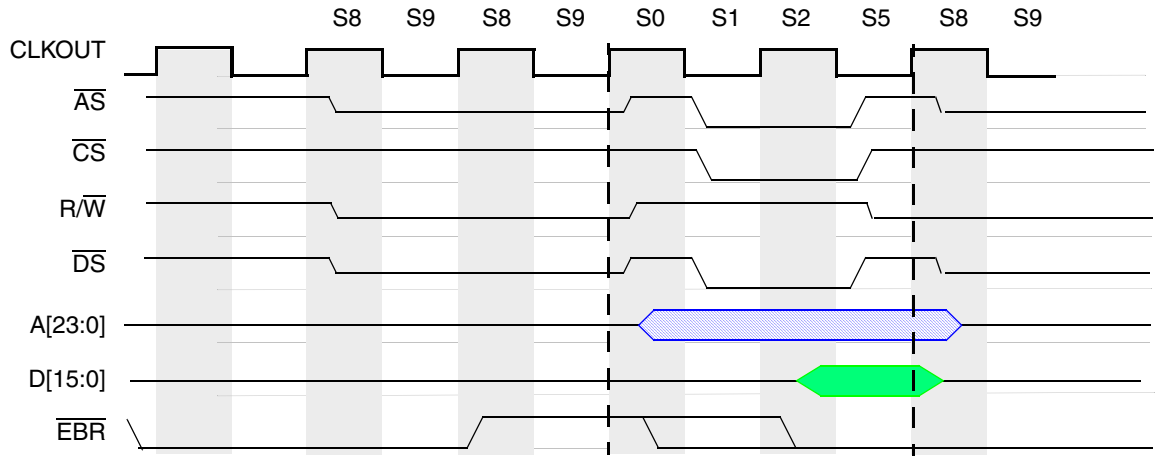


Figure 3-56 $\overline{\text{EBR}}$ Timing

SECTION 4
FASRAM

This document describes the fast access standby random access memory (FASRAM), an intermodule bus 3 (IMB3) module, designed to be the memory of a system in the modular embedded controller family of modular microcontrollers from Motorola. This family includes a series of modules from which numerous microcontrollers (MCUs) are being assembled. Currently available modules include CPU, timer, serial communications, system integration, and memory modules.

This memory module, in its fast access mode, currently works only with the CPU32X; however, other CPU modules which can use this memory will become available.

4.1 FASRAM Description

The FASRAM connects to both the IMB3 and the CPU of the MCU. To the IMB3, the FASRAM appears as a normal RAM memory module with normal (non-bursting) IMB3 timing. To the CPU, the private fast access bus (FAB) appears as a memory with half the access time of a normal IMB3 bus cycle, while permitting the CPU to simultaneously access another IMB3 module (via the IMB3) for instructions. The FASRAM is capable of arbitrating between the IMB3 and the FAB. When simultaneous accesses are requested, the FAB will have access priority.

The normal use of the FASRAM will be as stack space or as frequently accessed variable storage. The processor will also be able to execute code from the FASRAM. However, the instruction fetches must take place over the IMB3 since the CPU32x will dedicate the FAB to data accesses. These instruction fetches will not be burstable.

The FASRAM includes two comparators to facilitate debugging software, and a visibility bus (VB) that reflects the activity on the FAB.

4.2 FASRAM Features

- One clock fast access RAM for the attached CPU
- Two-clock access to the IMB3
- Long word, word, and byte accesses supported
- Low power and low voltage (3 V) design
- Two Kbytes expandable to 32 Kbytes in 2-Kbyte increments
- Variable size standby static RAM (mask option)
- Dynamic dual access (IMB3 and FAB) with operand coherency
- Two address comparators (equality or range comparisons)
- External visibility bus (as a bond-out option)

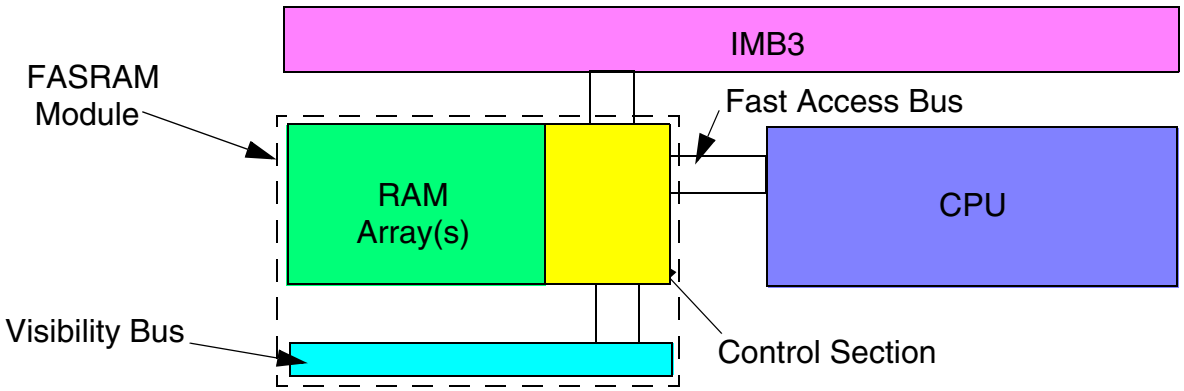


Figure 4-1 An MCU With a FASRAM Module

4.3 Operating Modes

The FASRAM module has five modes of operation: normal, standby, reset, test, and stop. After reset, the FASRAM is in stop mode.

4.3.1 Normal Operation

The normal operating mode for the FASRAM is to supply data to the IMB3 and the FAB whenever either bus requests an access. Priority is given to the FAB in case of simultaneous requests. The FAB access takes a minimum of one clock period, while the IMB3 access takes a minimum of two clock periods. Operating power is supplied by V_{DD} .

The memory can be read and written in bytes, words, or long words, by either bus. To guarantee coherency of operands, the FASRAM will refuse to grant access to one bus until the other completes its entire operand. Address acknowledge will be asserted to both sides upon a successful decode, however, the bus cycle to the “losing” side will have DTACK delayed until the “winning” side has finished. The maximum amount of data transferred for any single bus cycle is one word (16 bits). A byte or aligned word access takes only one bus cycle to complete. A long word or misaligned word access will take two bus cycles to complete.

The RAM array may be programmed to respond to either program space only or to both data and program space.The array may be programmed to respond to either supervisor space only or to both user and supervisor space. Each byte of the array appears at the same address and space for both an IMB3 access and a FAB access.

Operand coherency is provided by the IBLOCK signal and the FBLOCK signal. When either signal is asserted by a bus master during a RAM array access, the other bus master will not be granted access to the RAM array until the signal is negated. For the CPU32x, this signal is asserted only by the TAS instruction and during long word cycles. If a bus master does not use the IBLOCK signal or the FBLOCK signal, then the operand is not coherent; ISIZ and FSIZ do not affect coherency determination.

When the debugging comparators are in use and are forcing FAB accesses to be run on the IMB3, operand coherency can not be guaranteed.

For the initial configuration, the FASRAM will only perform data accesses across the FAB for the CPU32X. However, nothing in the design of this module shall preclude using the FAB for instruction accesses also. The FASRAM can still perform instruction accesses across the IMB3.

Access to the control registers is possible only over the IMB3. The FAB can only access the RAM array. The decoding for the module control register addresses will be local to this module.

An IMB3 access to the FASRAM will look exactly like an access to any other internal module. The IMB3 state machine in the FASRAM will assert the IMB3 handshaking signals with the required IMB3 timing.

The minimum V_{STBY} is 2.5 V and should not be greater than V_{DD} during normal operation. V_{STBY} should be equal to V_{DD} during the array current test.

4.3.2 Standby Operation

When the chip is to be powered down, the contents of the RAM array are maintained by the standby power supply, V_{STBY}. If the standby voltage falls below the minimum required voltage, then the RAM contents may be corrupted. The FASRAM will automatically switch to standby operation when the voltage on V_{DD} is below the voltage on V_{STBY}, with no loss of data. In this mode, the FASRAM does not respond to any bus cycles, and will set the stop bit in the FMCR. The system may experience unexpected operation when the CPU requests data from the FASRAM, since it will not be responding. If there is a bus cycle in progress during the time the power supply is switching, reads may be inaccurate and writes may corrupt the RAM contents. If standby operation is not desired, then the V_{STBY} pin should be connected to V_{DD}. The amount of RAM preserved in standby operation is determined by a mask option.

Differing implementations will have differing minimum amounts (and granularity) of standby RAM. The amount of current used by the V_{STBY} pin is proportional to the amount of RAM placed in standby mode.

Portions of the RAM array not selected by the mask option will not be preserved if V_{DD} falls below V_{STBY}. The current on V_{STBY} may exceed its specified maximum value at some time during the transition time that V_{DD} is at or below the voltage switch threshold to a threshold above V_{SS} . If the standby power supply cannot provide enough current to maintain V_{STBY} above the required minimum value, then a capacitor must be provided from V_{STBY} to V_{SS} The value of the capacitor may be calculated as C = I *t /V, where ‘I’ is the difference between the transition current requirement (approximately 1 mA per standby switch) and the power supply’s maximum current, ‘t’ is the duration of the V_{DD} transition near the voltage switch threshold. (The typical switching voltage is in the range $V_{DD} = V_{STBY} \pm V_T$).

4.3.3 Reset Operation

When a reset occurs, the FASRAM completes its current bus cycle before resetting. If a byte or word sized access was in progress, it will be completed. If a long word access was in progress, it is possible that only half (one word) of the operation will be completed. Reset will not directly affect the RAM array contents. (If reset is caused by power-on reset, then the parts of the RAM array not using VSTBY will of course be corrupted.)

System reset only affects the bus state machine and does not cause any bits to be reset in the control registers. Master reset will cause the control register bits to go to their reset state (as described in **4.7 Programmer's Model**). Most importantly, the FASRAM is forced into stop mode by a master reset.

4.3.4 Test Operation

Test mode is entered by asserting ITSTMODB from the test module. Many different tests may be run on the FASRAM by writing the appropriate bit(s) in the FTEST register while ITSTMOD is asserted. Test mode provides a way for the user to test the RAM, even after the MCU has been installed in a system. While in test mode (ITSTMODB asserted), the FASRAM may be used normally by a system, unless tests are actually in progress; however, running a test may corrupt the data in the RAM array.

When not in test mode, writes to the FTEST register will have no effect, and reads will always return zeros. All FASRAM tests must be conducted at a 3.3-V supply voltage and at room temperature. When performing the array current tests, operating power to the bit cells of the RAM array is provided by the VSTBY pin, while the VDD pin powers the control circuits. It is required that VSTBY is present when performing the array current tests. This facilitates measuring the array current.

4.3.4.1 Open Circuit Tests

[Description of test modes and operation of same.]

4.3.4.2 Array Current Tests

[Description of test modes and operation of same.]

4.3.4.3 Scan Tests

[Description of test modes and operation of same.]

4.3.5 Stop Operation

Setting the STOP bit of the FMCR causes the FASRAM to enter stop mode. In this mode, the RAM array cannot be accessed by either bus. All data in the RAM array and in the control registers is retained, and the module can still transition to standby mode. The control registers may still be accessed by the IMB3. A bus cycle (from either the IMB3 or the FAB) that would have gone to the RAM array is ignored, although FAB accesses continue to be shown on the VB (provided it is enabled). The debug comparators will not operate. External logic can then decode the RAM array access space.

Stop mode differs from standby mode in that stop mode is entered under software control (or forced by reset) and is independent of the power source. Stop mode is exited by clearing the STOP bit in the FMCR.

4.3.6 FREEZE Operation

When the CPU entered in the background mode, the CPU asserts IFREEZEB and the FASRAM completes its current cycle. In this mode, the FASRAM acts as same way as it is in test mode. When the background mode is entered, the RLCK bit of the FMCR may be cleared. The FTEST becomes writable, just like the FASRAM is in the test mode.

4.4 Visibility Bus

Because the IMB3 may be in use for instruction fetches, the FAB on the FASRAM module will not be visible to the development system (via a show cycle on the IMB3).

An extra visibility bus is available as a bond-out option. This visibility bus is unidirectional, and shows the address and data for the previous bus cycle on the FAB. The visibility bus will continue to operate even in stop mode, provided that it is enabled by setting the VEN bit in the FMCR. The frequency of access may be as great as once per clock cycle. The bus cycle on the FAB will be shown on the VB as soon as it completes. The VB data strobe for the bus cycle will be delayed from the completion of the FAB cycle by one-half or one clock.

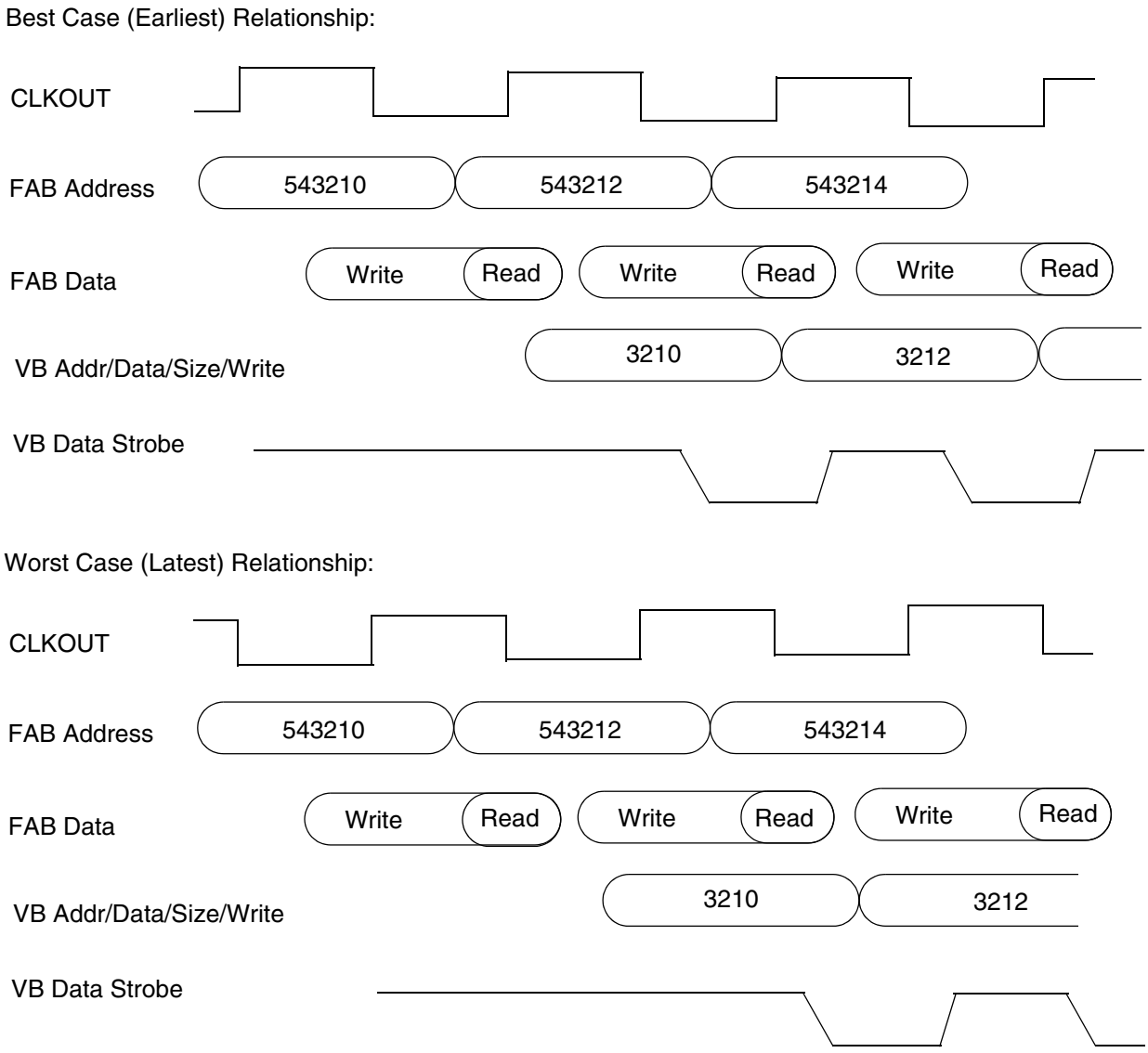


Figure 4-2 Timing Relationships Between the FAB and VB

The visibility bus will be driven to a new state shortly after the rising edge of CLKOUT, and the data strobe will be asserted shortly after the subsequent falling edge of CLK-OUT. If reset occurs before the VB has a chance to drive its pins, it is possible that the final FAB cycle will not be seen over the VB.

The visibility bus consists of up to 15 bits of the address decoded by the RAM array, and the data read or written to the RAM array. There is also a size pin, a read/write pin, and a data strobe. A bus cycle to any slave module on the FAB will appear on the visibility bus.

The visibility bus may be disabled by clearing the VEN bit in the FMCR. When the visibility bus is disabled (perhaps to reduce RFI), its address bus, data bus, and size pin are driven to their inactive state. The write pin and the data strobe will become comparator indicators. If a comparator is active, and it recognizes an in-range address, then the appropriate strobe will be asserted. The timing of the assertion of the comparator strobe will be the same as it would have been for a VB data strobe for that FAB cycle. To completely eliminate all activity on the visibility bus, you must disable the visibility bus and disable both comparators.

The write (VWRITE) pin and the data strobe (VDS) pins are a separate bond-out option. The rest of the visibility bus may be removed, while keeping these two pins.

4.4.1 Visibility Bus Address Pins (VADDR)

The portion of the address decoded by the RAM array is shown on the VADDR address bus. For example, if the array is six Kbytes in size, 13 address lines will be driven. If more than the necessary address lines are bonded out, the unused upper address lines should be ignored by the development system. Although the upper bits of the address are not shown, they are still used to match the base address of the RAM array as well as matching the comparator addresses. When the visibility bus is disabled, these pins are driven to a low state.

4.4.2 Visibility Bus Data Pins (VDATA)

The data for the RAM array access is driven on the VDATA data bus. A word sized access will drive all 16 bits of VDATA. When the visibility bus is disabled, these pins are driven to a low state.

If the access is byte sized, address bit 0 indicates which byte of the data bus contains valid data. If address bit 0 is one, then only data bits 7-0 are valid, and if address bit 0 is zero, then only data bits 15-8 are valid. The other byte should be ignored.

4.4.3 Visibility Bus Size Pin (VSIZE)

This VSIZE pin indicates the size of the RAM access. Word accesses will have a size of zero, and bytes will have a size of one. If the cycle is for the first word of a long word, a size indication of word is given. When the visibility bus is disabled, this pin is driven to a low state.

4.4.4 Visibility Bus Write Pin (VWRITE)

The direction of the RAM access is indicated on the VWRITE pin. If this pin is high, then the cycle was a read access. If this pin is low, then the cycle was a write access.

If the visibility bus is disabled, VWRITE becomes the comparator zero strobe. When comparator zero matches its address, this strobe will be asserted. Compares will be indicated for the FAB accesses when an action is taken (breakpoint is asserted over the FAB, or the FAB address acknowledge is inhibited). There is no indication of the address, data, size, or direction of the bus cycle.

4.4.5 Visibility Address Comparators

If the comparators are both set to range compare mode, then only VWRITE will be asserted on a compare; VDS will be asserted if the access is not in the range. For more details on range compares, see **4.5.3 Range Compare Mode**.

When the visibility bus and the comparators are all disabled, this pin is driven to a high state.

4.4.6 Visibility Bus Data Strobe (VDS)

When the address and data buses of the visibility bus have been driven to the value of the FAB address and data, then this strobe will be asserted. The development system will latch the contents of the visibility bus at the negative edge of this signal.

If the visibility bus is disabled, VDS becomes the comparator one strobe. When comparator one matches its address, this strobe will be asserted. Compares will be indicated for FAB accesses when an action is taken (breakpoint is asserted over the FAB, or the FAB address acknowledge is inhibited). There is no indication of the address, data, size, or direction of the bus cycle.

If the comparators are both set to range compare mode, then only VWRITE will be asserted on a compare; VDS will be asserted if the access is not in the range, provided that the address does reside in the FASRAM. For more details on range compares, see **4.5.3 Range Compare Mode**.

When the visibility bus and the comparators are all disabled, this pin is driven to a high state.

4.5 Address Comparators

Two address comparators are provided for development system support. These registers are capable of detecting a particular address during an access from the FAB.

The address may be optionally qualified by read or write, or user or supervisor mode. The comparators operate only during a RAM array access, never during a control register access. The comparators will not operate when the FASRAM is in stop mode. Bits 15 and 0 of the address are not used in any comparison. Both comparators may match on the same access. A comparator will not match an address if the address does not reside in the FASRAM.

When the comparators recognize that they should respond to an access, one of two actions may be taken. The CPU can be told that the address is not in the RAM array even if it actually is. Therefore, the CPU will be forced to run an IMB3 cycle to get the data. This IMB3 access may be seen externally if show cycles are enabled. This IMB3 access will be decoded by the FASRAM, and it will be treated just like any other cycle.

A forced IMB3 access will not cause the assertion of any MATCH bit in the FSTATUS register. Operand coherency cannot be guaranteed if the comparators force the bus cycle to the IMB3.

The other action that may be taken is to assert a breakpoint signal to the master. When the master runs an IMB3 breakpoint acknowledge cycle and a comparator has caused a breakpoint, then the FASRAM will respond to the breakpoint acknowledge cycle with a bus error signal. The breakpoint handler in the master should read the MATCH bits for each comparator in the FSTATUS register to determine which comparator caused the breakpoint, and then clear the asserted MATCH bit(s).

The comparators are enabled by setting at least one of the space bits (user or supervisor) and at least one of the read and write bits.

Each comparator has two basic modes of comparing addresses, equality and range. Equality mode matches an address if the access is within the word pointed to by the comparator address register. For range mode, comparator zero detects an address greater than or equal to the compare address, and comparator one detects an address less than or equal to the compare address. A comparator set to range mode matches an address only if the access is within the RAM array, never outside the RAM array.

The FASRAM will not make any response to IFREEZE if the CPU enters background mode in response to the breakpoint. The FASRAM will continue to operate normally.

The background mode program should negate the MATCH bits in the FSTATUS register to avoid confusing other software. If the visibility bus is disabled, then the VWRITE or VDS pins will be asserted for one-half clock period if their respective comparators match the current RAM array access address and function codes, in addition to performing the action specified in their control registers.

4.5.1 Equality Compare Modes

In equality compare mode, both comparators operate independently. The address of the bus cycle must exactly match the address in the compare register. Function codes and read or write are used to qualify the comparison. Bits 15 and 0 of the address are not used in the comparison.

If the comparator matches, then the selected action will be taken. If the action was to assert a breakpoint then the respective MATCH status bit will be set. The MATCH bit will not be set if the action was to force an IMB3 access.

If a comparator matches, and the visibility bus is disabled, then the appropriate strobe on the visibility bus will be asserted for one-half clock period. Comparator zero asserts VWRITE and comparator one asserts VDS.

4.5.2 Mixed Compare Modes

It is possible for one comparator to be in range compare mode, and the other to be in exact compare mode. In this mode, both comparators operate independently. The comparator configured for range mode will detect an access anywhere between the comparator address and the end of the RAM array. The direction of the range depends on which comparator is set to range mode. Comparator zero matches addresses equal to or greater than the comparator register, and comparator one matches addresses

less than or equal to the comparator register. Bits 15 and 0 of the address are not used in the comparison.

If the comparator matches, then the selected action will be taken. If the action was to assert a breakpoint then the respective MATCH status bit will be set. The MATCH bit will not be set if the action was to force an IMB3 access.

If a comparator matches, and the visibility bus is disabled, then the appropriate strobe on the visibility bus will be asserted for one-half clock period. Comparator zero asserts VWRITE and comparator one asserts VDS.

Setting a comparator to match on a range of addresses which are used for stack space is unwise. The CPU might be forced to take multiple breakpoint exceptions when only one was intended.

4.5.3 Range Compare Mode

In full range compare mode, when both comparators are in range mode, the comparator addresses are combined to form a range. The lower bound of the address range is in comparator zero, and the upper bound is in comparator one. Function codes and read or write are used to qualify the comparison. Bits 15 and 0 of the address are not used in the comparison. Since the individual comparators have separate qualification bits, it is strongly urged that both comparators be set to match on the same type of bus cycle, read or write and user or supervisor, or else unintended operation may occur.

When any access is made anywhere in the range, such that the access address is greater than or equal to the address for comparator zero and less than or equal to the address for comparator one, then the action specified by the comparator zero control bit will be performed. If the action was to assert a breakpoint then the MATCH status bit for comparator zero will be set. Only comparator zeros MATCH bit will be set on a range compare. The MATCH bit will not be set if the action was to force an IMB3 access. If the comparator range matches, and the visibility bus is disabled, then the VWRITE strobe on the visibility bus will be asserted for one-half clock period. VDS will be asserted on a non-compare. Therefore, when the visibility bus is disabled, any access to the FASRAM will result in the assertion of one of the two strobes.

If the comparator addresses are reversed, such that the lower bound is in comparator one and the upper bound is in comparator zero, then the comparators will never match an address. VDS will be asserted if the access resulted in only one comparator matching.

Setting the comparators to match on a range of addresses that are used for stack space is unwise. The CPU might be forced to take multiple breakpoint exceptions when only one was intended.

4.6 Usage

The following are a few examples of the procedures for operating the FASRAM.

4.6.1 Initialization

After a master reset, the FASRAM is in the stop mode. The control registers FMCR, FTEST, FCCR0, FCCR1, and FSTATUS have been forced to the reset values as specified in **4.8.1 FASRAM Module Configuration, Control, and Status Registers**. Control registers FBAR, comparator values zero and one, FMATCH, and some bits of FSTATUS are in a random state, even if standby power has been on. (Standby power supports only the RAM array.) In order to activate the FASRAM, do the following:

1. Write the FBAR registers to the value used for the base address of the RAM array.
2. Write the RLCK bit in the FMCR to a one. No subsequent instruction will be able to change the FBAR registers. This may be done at the same time as step three.
3. Write the STOP bit in the FMCR to a zero. This enables the FASRAM.

There are several other ‘housekeeping’ things that may be done now. The array space bits, the delay bit, and enable the visibility bus may be configured.

4.6.2 Comparator Setup

The comparators are off after a master reset. In order to turn them on for range compares, do the following:

1. Set the visibility bus to be on or off. If the VB is off, then when the comparators are enabled, the VDS and VWRITE pins become comparator strobes.
2. Write the range’s low address to comparator zero, and the high address to comparator one.
3. Set both comparators to range mode. This may be done at the same time as step five.
4. Set comparator zero to perform the desired action, either a breakpoint or a forced IMB3 cycle. This may be done at the same time as step five.
5. Set both comparators to activate on the same type of cycle. The read, write, user, and supervisor bits should be the same for both comparators. This action actually enables the comparators.

Now the comparators are set to perform range compares. It is also possible to do exact compares by clearing the range bits. In this case, the bits indicating the cycle type do not have to be the same for each comparator.

4.6.3 Servicing the Comparators

If the comparators have asserted a breakpoint, then the CPU may or may not run a breakpoint acknowledge cycle. If it does, and the FASRAM has actually asserted a breakpoint, then the FASRAM will terminate the breakpoint acknowledge cycle with a bus error. This should cause the CPU to perform its breakpoint handler routine. The breakpoint handler should do the following:

1. Read the MATCH bit in the FSTATUS register. If set, then this is probably the comparator that caused the breakpoint. If both MATCH bits are set, you must use the WHICH bit or the saved address to tell which comparator actually

caused the most recent breakpoint.

2. Write a zero to the MATCH bit. This clears the indication of a breakpoint. It is permissible to write a zero even if it is already cleared. Steps one and two may be done with a BCLR instruction on the CPU32x.
3. If the only interest is in this breakpoint address for a single access, then turn off the comparator by writing zeros to the comparator control register.
4. If needed, read the saved breakpoint address and the associated information.
5. Check all other sources of breakpoints, including the other comparator. It is possible for multiple breakpoint sources to result in a single breakpoint exception. Software must decide what to do in that situation.
6. If the CPU runs a breakpoint acknowledge cycle, but neither MATCH bit is set, then the FASRAM will not respond to the cycle.

4.6.4 Testing the FASRAM

To test the FASRAM:

1. Enable test mode for all modules. Consult the integration module specification for guidelines.
2. Write to the test block of the integration module, and enable test mode. The RLCK bit can now be cleared.
3. Write appropriate commands to the FTEST register in the FASRAM.

Be certain that no attempt is made to access the FASRAM in the middle of a test. The FASRAM is not defined to operate correctly while a test is in progress.

4.7 Programmer's Model

The FASRAM module consists of two separately addressable sections, the RAM array itself and the control and status registers. The address mapping of the RAM array is governed by the control registers. Once the RLCK bit is set, the base address of the array is locked and cannot be re-mapped, unless the RLCK bit is cleared either in test mode or background mode. The control registers are mapped to a space within the 32-Kbyte module control block starting at 0x007FFB00 or 0xFFFFFB00 depending on the modmap signal on the IMB3, within supervisor data space. The mapping within the module control block is mask programmable.

All of the programmer's control and status registers are contained in the control section and are accessible via the IMB3 only. This includes any debugging or test features.

The FASRAM memory map occupies 32 bytes. **Figure 4-3** shows the decoding of the address lines for the FASRAM internal registers. A[6:0] must match the individual register decodes as detailed in **Table 4-1**.

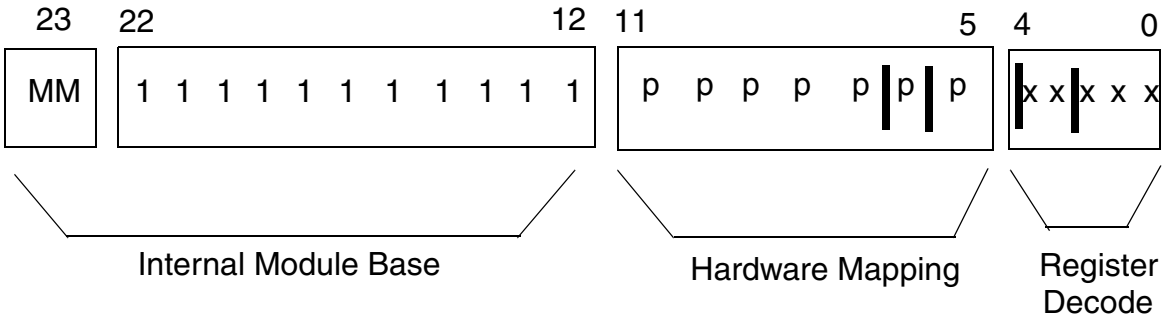


Figure 4-3 FASRAM Module Address Decoding

Table 4-1 shows all of the control registers as well as the RAM array. The registers are logically grouped as those registers needed to affect:

1. Array address mapping
2. Testing features
3. Debugging features

4.8 FASRAM RAM Array Addressing / Mapping

The RAM array can be placed anywhere in the address map of the MCU by means of the FASRAM base address register (FBAR) provided the following conditions hold:

1. The base address of the RAM array must be an even multiple of the RAM array size rounded up to the next power of two. Therefore, while the FASRAM may be im-plemented with RAM array sizes from two to 32 Kbytes in increments of two Kbytes, the lowest address within the array, the base address, must be on a 2-, 4-, 8-, 16-, or 32-Kbyte boundary. Only those address lines which define placement of the block will be compared (e.g., for an 8-Kbyte FASRAM only FBAR bits A[23:13] will be compared). This is mask programmable and based on the size of the implemented RAM array.
2. The RAM array also must not overlap any part of the 4-Kbyte internal peripheral register area, within user or supervisor data space. In the event that this occurs, the RAM array will not be accessible, allowing the control registers to be accessible so that they can be modified.
3. The FBAR registers may be written multiple times after reset, but will be locked once the RLCK bit in the FMCR is set. The FASRAM must be in stop mode and RLCK cleared for the FBAR registers to be writable. Once RLCK is set, the FBAR registers may not be altered and RLCK may not be cleared until master reset is asserted. This prevents accidental re-mapping of the array. An exception to this functionality is that when either test mode or background mode is entered, RCLK may be cleared. For more information on this operation, see 4.3.4 Test Operation.

Table 4-1 FASRAM Module Programmers Model

Offset Address from the Control Register Mapping Address	FASRAM Module Control Registers	
	15 8	7 0
0xYF F6C0	Module configuration register (FMCR)	
0xYF F6C2	Test register (FTEST)	
0xYF F6C4	Base address (FBAR-H)	
0xYF F6C6	(FBAR-L)	
0xYF F6C8	Comparator 0 (≥) value FCMP0)0	
0xYF F6CA	Comparator 0 (≤) value FCMP1)0	
0xYF F6CC	Comparator control 0 (FCCR0)	Comparator control 1 (FCCR1)
0xYF F6CE	Most recent match address (FMATCH)	
0xYF F610	Not Used[15:8] (Reserved)	FSTATUS[7:0]
0xYF F612	Not used (Reserved)	
—	—	
0xYF F61E	Not used (Reserved)	
FASRAM Module RAM Array		
ADDRESS	15 8	7 0
*___*0000	—	
*___*0002	—	
—	—	
*___*FFFE	—	

NOTES:
1. Yellow areas indicate locations which are *not used* and are *reserved* for future use. *Reserved* locations cannot be modified and alsways return 0's when read. No addresses are labeled *unused*.
2. Implemented memory array size can vary from 2 Kbytes to 32 Kbytes in 2-Kbyte increments.

The register referred to as "FBAR" is actually the FASRAM base address registers FBAR-H and FBAR-L, concatenated to form a long word. These may be treated as two distinct word sized registers.

For each bus cycle from the IMB3 or the FAB, the high order address lines A[23:N] (where N = 11, 12, 13, 14, 15 for a FASRAM of size rounded up to 2, 4, 8, 16, and 32 Kbytes) are compared with the value of the FBAR registers. The function codes are also qualified with the RASP bits in the FMCR for restrictions on the type of access allowed. If these values match, then the low order address lines and the SIZ[1:0] lines are latched and used to access the array.

The FASRAM will only respond to addresses which are within the implemented RAM array size.

EXAMPLE

For a 10-Kbyte FASRAM implementation, FBAR can be set to any address which is a multiple of 16 Kbytes. The RAM array will then be mapped to the addresses which fall between the base address and the base address plus 10 Kbytes. In this example, the FASRAM will not respond to the addresses from the top of the 10-Kbyte RAM array to the next 16-Kbyte boundary.

Accesses to the FASRAM RAM array operate identically, except for bus cycle timing differences, for accesses from both the IMB3 and the FAB. For both buses, the memory resides in the same function code space and at the same address.

4.8.1 FASRAM Module Configuration, Control, and Status Registers

All configuration and control of the FASRAM module can be accomplished through the registers described in this section. To protect these registers from accidental modification, they are mapped to supervisor data space only. In addition, they can be accessed only via the IMB3 and not the FAB. The FASRAM registers will not respond to a user data space access, or a program space access, therefore any user-mode or program-mode access of the registers will be acknowledged by other modules on the chip or by an external device.

While at present only 18 bytes have been used for the control registers, this module decodes a 32-byte register block. Unimplemented registers are read as zeros and writes have no effect.

All of the FASRAM control registers can be accessed by byte-sized or word-sized bus cycles. There are no word-sized-only registers.

The following sections describe the operation of each register in the FASRAM control block. Any restrictions to making changes to the registers will be noted in the description of the register. Unless otherwise noted, a master reset will cause register bits to be forced to their reset state, while a system reset will have no effect on the registers.

4.8.1.1 FASRAM Module Configuration Register (FMCR)

All modules on the IMB3 contain a module configuration register. The FMCR bits configure the FASRAM module for stop operation and for proper access rights to the RAM array. They also control the functionality of the V_{STBY} power and visibility bus (VB). Register bits which are labeled with a “0” are reserved for future use. Reserved locations cannot be modified and always return zeros when read.

FMCR — FASRAM Module Configuration Register **0xYF F6C0**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
STOP	VEN	RESERVED		RLCK	DLY1	RASP		Reserved							
RESET:															
1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0

Table 4-2 FMCR Bit Descriptions

Bit(s)	Name	Description
15	STOP	Stop control. Setting the STOP control bit in the FMCR register forces the FASRAM module to enter into the STOP state. When the STOP bit is set, RAM array accesses are ignored. When the STOP bit is cleared, the FASRAM array base address register (FBAR) is write protected. This bit is automatically set when VDD is less than VSTBY, and during the array current test. 0 = FASRAM module is in normal mode of operation 1 = Causes the FASRAM module to enter the low power stop mode
14	VEN	Visibility bus enable. The visibility bus control signals VDS and VWRITE are reconfigured as debug comparator strobe out-put signals. The visibility bus enable bit enables the external visibility bus (VB) to show accesses from the fast access bus (FAB) to the FASRAM RAM array. For more information, see **4.4 Visibility Bus**. When the VB is disabled, the VWRITE and VDS lines act as comparator strobes for the two comparators. For the comparator strobe to be generated, one or both of the comparators must be enabled. For more information, see **4.4.4 Visibility Bus Write Pin (VWRITE)** and **4.4.6 Visibility Bus Data Strobe (VDS)**. 0 = Visibility bus (VB) is driven to an inactive level (V_{SS} or V_{DD}) 1 = Visibility bus (VB) shows the accesses on the fast access bus (FAB)
13:12	—	Reserved
11	RLCK	RAM base address lock. The FASRAM base address registers are writable until the RLCK bit is set. Once RLCK is set, writes can not affect the base address registers. This bit may only be set via the IMB3 and is reset only by a master reset. Once set, this bit may be cleared by software only when the FASRAM is either in test mode or in background mode. This bit may be written any number of times with a ‘0’ until it is written with a ‘1’. 0 = FASRAM base address registers are writable from the IMB3 1 = FASRAM base address registers are write locked
10	DLY1	Bus cycle delay. The FASRAM uses this bit to increase internal array access times by one internal clock (one-half of a system clock). This causes IMB3 cycles to delay completion by one system clock, and FAB cycles to delay completion by one-half of a system clock. This feature may be required depending on supply voltages and memory array implementation. 0 = FASRAM accesses are normal speed 1 = FASRAM accesses are delayed by one-half system clock
9:8	RASP	FASRAM array space. The RAM array is placed either in supervisor or in unrestricted space. When placed in supervisor space, (RASP[1] = “1”), only a supervisor program can access the array. If a user program is attempting to access the FASRAM while it is in supervisor space, the FASRAM will ignore the access. If RASP[1] = “0”, the RAM array is placed in unrestricted space and accesses by both supervisor and user programs are allowed. The RAM array is placed either in program or in program/data space. When placed in program space, (RASP[0] = “1”) only program instructions, or program counter relative addressing modes for operand fetches can be read from the array. All data space accesses to the RAM array will be ignored. If RASP[0] = “0”, the RAM array will respond to both program and data space accesses. Accesses from both the FAB and the IMB3 are affected by the RASP bits. The RASP bits have no effect on control register accesses.. 00 = Unrestricted program and data 01 = Unrestricted program 10 = Supervisor program and data 11 = Supervisor program
7:0	—	Reserved

4.8.2 Array Base Address Registers (FBAR)

The RAM array base address registers (FBAR-H and FBAR-L, referred to collectively as FBAR) are provided to allow the flexibility of placing the RAM array anywhere in the memory map. The FBAR contains an address field used to specify the most significant bits of the lowest addressable value in the RAM array address block. The lower eleven

bits of FBAR (FBAR[10:0]) and the upper eight bits of FBAR (FBAR[31:24]) read as '0' and are not affected by writes.

The restrictions on the mapping of the RAM array are detailed in **4.8 FASRAM RAM Array Addressing / Mapping**.

FBAR-H — Base Address Register **0xYF F6C4**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
Reserved								BAR [A23:A16]							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FBAR-L — Base Address Register **0xYF F6C6**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
BAR [A15:A11]					Reserved										
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-3 FBAR-H, FBAR-L Bit Descriptions

Bit(s)	Name	Description
FBAR-H 15:8, FBAR-L 10:0	—	Reserved
FBAR-H 7:0, FBAR-L 15:11	BAR [A31:A11]	The RAM array's base address is contained in the base address field of the array base address registers. With STOP set and RLCK cleared the base address field of FBAR may be changed so that the array may be placed at the desired address in the memory map. This must be done by a supervisor program since the registers are in supervisor data space. To lock the base address field RLCK should be set. This will prevent the base address field from being changed until the next master reset. Once RLCK has been set, it cannot be cleared except either in test mode or in background mode. Supervisor read from IMB3, writable from IMB3 if STOP = 1 and RLCK = 0.

4.9 FASRAM Debugging Features

The FASRAM has a set of control registers in addition to those contained in the standard SRAM module. The purpose of these registers is to control the debugging and development features found in the FASRAM module.

The debugging features consist of two comparators which can be used to detect accesses to addresses within the FASRAM in one of three manners:

- To trigger on two specific addresses
- To trigger on a range of addresses by specifying the top and bottom of the range
- To trigger on one specific address and one range which is bounded by the memory border

One comparator is a greater than or equal to function while the other is a less than or equal to function. The comparators can be used together in their RANGE mode to detect accesses within a range, or they may be used separately to detect an access to one of two specific addresses.

There are also two status-only registers associated with the comparators. The two registers show information relevant to the most recent match address which was detected by either comparator. There is a most recent match address register and a status register.

The FASRAM debugging comparators are disabled after master reset. It should be noted that enabling the comparators will impact system performance in two ways:

1. Processor performance will be somewhat impaired due to breakpoints and IMB3 data accesses where they would otherwise not occur. This can be kept to a minimum by using the debugging comparators wisely, limiting their scope only to vital areas of the FASRAM RAM array space.
2. Slightly more power will be consumed when one or more comparators are enabled.

4.9.1 Comparator Control Registers (FCCR0, FCCR1)

There are two control registers, one for each of the two comparators.

The comparator control registers also contain four bits which control when a compare is to take place, a bit which controls what type of compare to do (range or equals), and a bit which controls the action which will occur when a compare triggers. The comparator is turned off by clearing all four of the control bits. This is the reset state of the control registers. The comparator is effectively disabled when the control bits are cleared. The comparator control registers (FCCR0, FCCR1) shows the position and reset states of these bits.

FCCR0, FCCR1— FASRAM Comparator Value 0,1 Registers **0xYF F6C8**
0xYF F6CA

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
Comparator 0								Comparator 1							
	RANGE	FRCI	USER	SUPV	READ	WRITE		0	0	RANGE	FRCI	USER	SUPV	READ	WRITE
RESET: S															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

One or both of the comparators should be enabled by writing to the comparator control registers. At least one of the READ or WRITE bits as well as at least one of the USER or SUPV bits must be set to enable a given comparator. These bits determine what type of access will trigger the comparator. Register bits which are labeled with a “0” are reserved for future use. Reserved locations cannot be modified and always return zeros when read.

Table 4-4 FCCR0, FCCR1 Bit Descriptions

Bit(s)	Name	Description
13, 5	RANGEI	Type of compare to perform. The RANGE control bits set a specific comparator to the range compare mode. This means that the comparator will now “trigger” when an access is “≤” or “≥” depending on which comparator is used. If RANGE = “0” then the comparator will act as an “==” comparator. When the RANGE bits for both comparators are set, the two comparators work together as a range detection where an access must “trigger” both comparators to cause an action. This is useful when the user wishes to detect accesses between two arbitrary addresses within the FASRAM, neither of which is a RAM array boundary. If only one of the comparators’ RANGE bits is set then the associated comparator must be used to specify a range of addresses whose other bound is the end of the RAM array.
12, 4	FRCI	Setting actions resulting from a compare ‘trigger’. The debugging comparators have two possible actions they can perform on an access to the FASRAM. 0 = Comparators cause a breakpoint to occur on the memory access 1 = Force access to be performed via the IMB3
11, 3	USER	Compare on a user access. These control bits enable the specific comparator to detect a user access to a FASRAM RAM array location via the FAB. SUPV may also be set. READ and / or WRITE must be set.
10, 2	SUPV	Compare on a supervisor access. These control bits enable the specific comparator to detect a supervisor access to a FASRAM RAM array location via the FAB. USER may also be set. READ and / or WRITE must be set.
9, 1	READ	Compare a read access. These control bits enable the specific comparator to detect a read access to a FASRAM RAM array location via the FAB. WRITE may also be set. SUPV and / or USER must be set.
8, 0	WRITE	Compare a write access. These control bits enable the specific comparator to detect a write access to a FASRAM RAM array location via the FAB. READ may also be set. SUPV and / or USER must be set.

4.9.2 Finding the Status of the Most Recent Compare

The FASRAM has two status registers which give the user information about the most recent comparator “trigger” which was generated. This information includes the following:

- Which comparator had the most recent match (WHICH)
- Which comparator had a match (MATCH1, MATCH0)
- The size of the access (SIZE)
- Was the access a read or a write (RWSTAT)
- Was the access from data or program space (DPSTAT)
- Was the access from supervisor or user space (SUSTAT)
- The actual address which caused the “trigger” (most recent match address)

4.9.3 Most Recent Match Address (FMATCH)

The most recent match address contains the address of the last match made by either comparator. The FMATCH register is affected when a breakpoint is caused, not for a forced IMB3 cycle. Although bits 15 and 0 are not used in the address comparison, they are captured in this register. The WHICH bit in the STATUS register indicates which comparator made the most recent match. This register is read-only.

If both comparators trigger before the FMATCH register can be read, the value in the FMATCH register will be for the last trigger. This may lead to software becoming confused as to which comparator the FMATCH register represents.

Reset has no effect on this register.

FMATCH — Most Recent Match Address Register **0xYF F6CE**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESET: S															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

4.9.4 Most Recent Match Status (FSTATUS)

The STATUS register contains all pertinent information about the most recent comparator match, if and only if that match caused a breakpoint. All bits in the FSTATUS register are read-only with the following exception. A MATCH bit (FSTATUS[6:5]) is clearable by writing it to a zero. This can only be done following a read to FSTATUS (address YMMFFB11). A read to the adjacent byte “not used[15:8] (reserved)” (address YMMFFB10) will have the same affect. A write of a one has no effect. Most status bits in the FSTATUS register have no reset state, with the exception that the MATCH bits are cleared. FSTATUS[15:8] and FSTATUS[4] are unused and reserved. This means writes have no effect and reads always return zeros.

FSTATUS— FASRAM Most Recent Match Status Register **0xYF F610**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
Reserved								WHICH	MATCH1	MATCH0	Reserved	SIZE	RWSTAT	DPSTAT	SUSTAT
RESET: S															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-5 FSTATUS Bit Descriptions

Bit(s)	Name	Description
15:8	—	Reserved
7	WHICH	Most recent comparator match. The WHICH bit indicates which of the two comparators most recently had a match, provided a breakpoint action was caused. The WHICH bit is cleared if comparator zero had a match and is set if comparator one had a match. The WHICH bit is undefined if both RANGE bits are set, or if both comparators trigger on the same access.
6:5	MATCH1, MATCH0	Compare on a user access. The MATCH bits in the FSTATUS register indicate whether the associated comparator has been “triggered”. The MATCH bits are only set when a comparator trigger causes a breakpoint (FCRI = 0). Once these bits are set via a compare trigger, they will remain set until cleared by a master reset or by a supervisor write clearing these bits in the FSTATUS register. These bits may be written to a zero only if they have been read while set to one. This allows for proper determination of which comparator has breakpointed when the second comparator causes a breakpoint after the register has been read for a first breakpoint.
4	—	Reserved
3	SIZE	Size of the access. The SIZE bit in the FSTATUS register indicates the size of the most recent access for which the comparator has a match, provided a breakpoint action was caused. 0 = Access was a word access 1 = Access was a byte access
2	RWSTAT	Access a read or a write. The RWSTAT bit indicates whether the most recent comparator match was caused by a read or by a write access, provided a breakpoint action was caused. 0 = Match was caused by a write 1 = Match was caused by a read
1	DPSTAT	Access from data or program space. The DPSTAT bit indicates whether the most recent comparator match was caused by a data (or by a program space access, provided a breakpoint action was caused. The CPU32x is not capable of making a program space access over the FAB, therefore this bit will always be a one for that CPU. 0 = Most recent comparator match was caused by a program space access 1 = Most recent comparator match was caused by a data space access
0	SUSTAT	Access from data or program space. The SUSTAT bit indicates whether the most recent comparator match was caused by a supervisor or by a user space access, provided a breakpoint action was caused. 0 = Most recent comparator match was caused by a user access 1 = Most recent comparator match was caused by a supervisor space access

4.10 Bus Connections

The following paragraphs describe the various bus connections to the FASRAM.

4.10.1 Inter-Module Bus

The intermodule bus (IMB3) is an internal, bi-directional two-clock bus with the data for one cycle overlapping the address for the next cycle. The FASRAM will respond to an IMB3 request in a minimum of two clock periods, but may be slower if the FAB is currently accessing the RAM array.

4.11 Fast Access Bus

The fast access bus (FAB) is an internal, bidirectional one-clock interface to the CPU. It is a miniature IMB3. The FASRAM will respond to a FAB request in a minimum of one clock period, but may be slower if the IMB3 is currently accessing the FASRAM.

4.12 Visibility Bus

The visibility bus (VB) is the output-only bus that shows the external world the activity on the fast access bus. There are three bond-out options: the entire VB, just the write pin and the data strobe, or none.

The address and data for the FAB cycle are buffered internally until the bus cycle is complete. Then the VB is driven and the strobes asserted. This eliminates any variable timing due to the differences of the data availability between a read cycle and a write cycle, or between the address and data. No indication of a long cycle is given.The actual pin names and number of implemented bits are specified in the chip plan. See Table 4-6.

Table 4-6 Visibility Bus Pins

Pin	I/O	Size	Active	Function
VADDR (14:0)	O	15[1]	N/A	Address of the FAB cycle. Number of implemented pins depends on total memory size.
VDATA (15:0)	O	16	N/A	Data for the FAB cycle.
VSIZE	O	1	N/A	Size of the FAB cycle. Word = 0, Byte = 1
VWRITEB or VWRITE	O	1	Low	Direction of the FAB cycle. Read = 1, Write = 0. If the VB is disabled, then this is the comparator zero strobe.
VDSB or VDS	O	1	Low	External logic should latch the VB. If the VB is disabled, then this is the comparator one strobe.

NOTE

1. The number of address pins depends on the size of the RAM array. For example, 13 pins would be used for an 8-Kbyte array.

4.13 Electrical Characteristics

For electrical characteristics, please refer to APPENDIX E ELECTRICAL AND AC CHARACTERISTICS.

SECTION 5
TIME PROCESSOR UNIT 3

There are two time processor unit 3's (TPU3). This enhanced version of the original TPU is an intelligent, semi-autonomous microcontroller designed for timing control. The TPU3 is fully compatible to the TPU2. Operating simultaneously with the CPU, the TPU3 modules process micro-instructions, schedules and processes real-time hardware events, performs input and output, and accesses shared data without CPU intervention. Consequently, for each timer event, the CPU setup and service times are minimized or eliminated. **Figure 5-1** is a simplified block diagram of the TPU3.

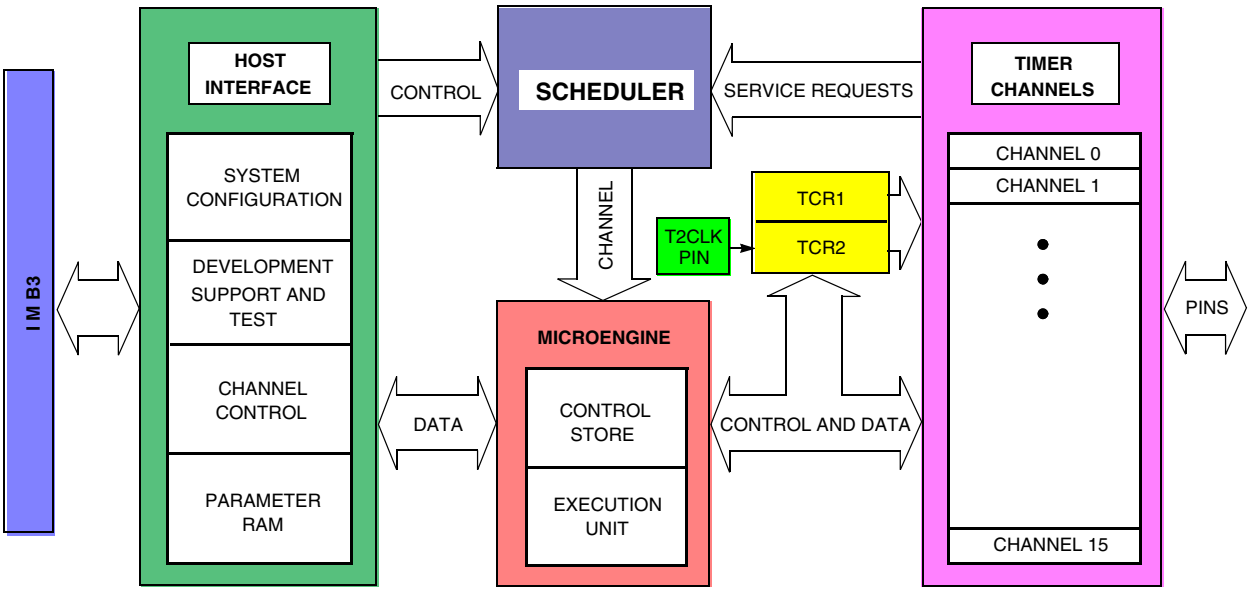


Figure 5-1 TPU3 Block Diagram

5.1 Overview

The TPU3 can be viewed as a special-purpose microcomputer that performs a programmable series of two operations, match and capture. Each occurrence of either operation is called an event. A programmed series of events is called a function. TPU functions replace software functions that would require CPU interrupt service.

The microcode ROM TPU3 functions that are available in the MC68377 are described in **APPENDIX D TPU ROM FUNCTIONS**.

5.2 TPU3 Components

Each TPU3 consists of two 16-bit time bases, 16 independent timer channels, a task scheduler, a microengine, and a host interface. In addition, a dual-ported parameter RAM is used to pass parameters between the module and the CPU.

5.2.1 Time Bases

Two 16-bit counters provide reference time bases for all output compare and input capture events. Prescalers for both time bases are controlled by the CPU via bit fields in the TPU3 module configuration register (TPUMCR) and TPU module configuration register two (TPUMCR2). Timer count registers TCR1 and TCR2 provide access to the current counter values. TCR1 and TCR2 can be read by TPU microcode but are not directly available to the CPU. The TCR1 clock is always derived from the system clock. The TCR2 clock can be derived from the system clock or from an external input via the T2CLK clock pin. The duration between active edges on the T2CLK clock pin must be at least nine system clocks.

5.2.2 Timer Channels

The TPU3s have 16 independent channels, each connected to an MCU pin. The channels have identical hardware and are functionally equivalent in operation. Each channel consists of an event register and pin control logic. The event register contains a 16-bit capture register, a 16-bit compare/match register, and a 16-bit greater-than-or-equal-to comparator. The direction of each pin, either output or input, is determined by the TPU microengine. Each channel can either use the same time base for match and capture, or can use one time base for match and the other for capture.

5.2.3 Scheduler

When a service request is received, the scheduler determines which TPU3 channel is serviced by the microengine. A channel can request service for one of four reasons: for host service, for a link to another channel, for a match event, or for a capture event. The host system assigns each active channel one of three priorities: high, middle, or low. When multiple service requests are received simultaneously, a priority-scheduling mechanism grants service based on channel number and assigned priority.

5.2.4 Microengine

The microengine is composed of a control store and an execution unit. Control-store ROM holds the microcode for each factory-masked time function. When assigned to a channel by the scheduler, the execution unit executes microcode for a function assigned to that channel by the CPU. Microcode can also be executed from the dual-port RAM (DPTRAM) module instead of the control store. The DPTRAM allows emulation and development of custom TPU microcode without the generation of a microcode ROM mask. Refer to **5.3.6 Emulation Support** for more information.

5.2.5 Host Interface

The host interface registers allow communication between the CPU and the TPU3, both before and during execution of a time function. The registers are accessible from the IMB through the TPU3 bus interface unit. Refer to **5.4 Programming Model** for register bit/field definitions and address mapping.

5.2.6 Parameter RAM

Parameter RAM occupies 256 bytes at the top of the system address map. Channel parameters are organized as 128 16-bit words. Channels zero through 15 each have eight parameters. The parameter RAM address map in **5.4.13 TPU3 Parameter RAM** shows how parameter words are organized in memory.

The CPU specifies function parameters by writing to the appropriate RAM address. The TPU3 reads the RAM to determine channel operation. The TPU3 can also store information to be read by the CPU in the parameter RAM. Detailed descriptions of the parameters required by each time function are beyond the scope of this manual. Refer to Motorola's ***TPU Literature Package*, TPULITPAK/D,** for the available TPU documentation.

5.3 TPU Operation

All TPU3 functions are related to one of the two 16-bit time bases. Functions are synthesized by combining sequences of match events and capture events. Because the primitives are implemented in hardware, the TPU3 can determine precisely when a match or capture event occurs, and respond rapidly. An event register for each channel provides for simultaneous match/capture event occurrences on all channels.

When a match or input capture event requiring service occurs, the affected channel generates a service request to the scheduler. The scheduler determines the priority of the request and assigns the channel to the microengine at the first available time. The microengine performs the function defined by the content of the control store or emulation RAM, using parameters from the parameter RAM.

5.3.1 Event Timing

Match and capture events are handled by independent channel hardware. This provides an event accuracy of one time-base clock period, regardless of the number of channels that are active. An event normally causes a channel to request service. The time needed to respond to and service an event is determined by which channels and the number of channels requesting service, the relative priorities of the channels requesting service, and the microcode execution time of the active functions. Worst-case event service time (latency) determines TPU3 performance in a given application. Latency can be closely estimated. Refer to Motorola's ***TPU Literature Package*, TPULITPAK/D,** for the available TPU documentation.

5.3.2 Channel Orthogonality

Most timer systems are limited by the fixed number of functions assigned to each pin. All TPU3 channels contain identical hardware and are functionally equivalent in oper-

ation, so that any channel can be configured to perform any time function. The user controls the combination of time functions.

5.3.3 Interchannel Communication

The autonomy of the TPU3 is enhanced by the ability of a channel to affect the operation of one or more other channels without CPU intervention. Interchannel communication can be accomplished by issuing a link service request to another channel, by controlling another channel directly, or by accessing the parameter RAM of another channel.

5.3.4 Programmable Channel Service Priority

The TPU3 provides a programmable service priority level to each channel. Three priority levels are available. When more than one channel of a given priority requests service at the same time, arbitration is accomplished according to channel number. To prevent a single high-priority channel from permanently blocking other functions, other service requests of the same priority are performed in channel order after the lowest-numbered, highest-priority channel is serviced.

5.3.5 Coherency

For data to be coherent, all available portions of the data must be identical in age, or must be logically related. As an example, consider a 32-bit counter value that is read and written as two 16-bit words. The 32-bit value is read-coherent only if both 16-bit portions are updated at the same time, and write-coherent only if both portions take effect at the same time. Parameter RAM hardware supports coherent access of two adjacent 16-bit parameters. The host CPU must use a long-word operation to guarantee coherency.

5.3.6 Emulation Support

Although factory-programmed time functions can perform a wide variety of control tasks, they may not be ideal for all applications. The TPU3 provides emulation capability that allows the user to develop new time functions. Emulation mode is entered by setting the EMU bit in TPUMCR. In emulation mode, an auxiliary bus connection is made between the DPTRAM and the TPU3, and access to DPTRAM via the intermodule bus is disabled. A 9-bit address bus, a 32-bit data bus, and control lines transfer information between the modules. To ensure exact emulation, DPTFLASH module access timing remains consistent with access timing of the TPU microcode ROM control store.

To support changing TPU application requirements, Motorola has established a TPU function library. The function library is a collection of TPU functions written for easy assembly in combination with each other or with custom functions. Refer to Motorola's ***TPU Literature Package*, TPULITPAK/D,** for the available TPU documentation.

5.3.7 TPU3 Interrupts

Each of the TPU channels can generate an interrupt service request. Interrupts for each channel must be enabled by writing to the appropriate control bit in the channel interrupt enable register (CIER). The channel interrupt status register (CISR) contains one interrupt status flag per channel. Time functions set the flags. Setting a flag bit causes the TPU to make an interrupt service request if the corresponding channel interrupt enable bit is set and the interrupt request level is non-zero.

The value of the channel interrupt request level (CIRL) field in the TPU interrupt configuration register (TICR) determines the priority of all TPU interrupt service requests. CIRL values correspond to MCU interrupt request signals IRQ[7:1]. IRQ7 is the highest-priority request signal; IRQ1 has the lowest priority. Assigning a value of 0b111 to CIRL causes IRQ7 to be asserted when a TPU interrupt request is made; lower field values cause corresponding lower-priority interrupt request signals to be asserted. Assigning CIRL a value of 0b000 disables all interrupts.

The CPU32 recognizes only interrupt requests of a priority greater than the value contained in the interrupt priority (IP) mask in the CPU status register. When the CPU32 acknowledges an interrupt request, the priority of the acknowledged interrupt is written to the IP mask and is driven out onto the IMB address lines.

When the IP mask value driven out on the address lines is the same as the CIRL value, the TPU contends for arbitration priority. The IARB field in TPUMCR contains the TPU arbitration number. Each module that can make an interrupt service request must be assigned a unique non-zero IARB value in order to implement an arbitration scheme. Arbitration is performed by means of serial assertion of IARB field bit values. The IARB of TPUMCR is initialized to 0x0 during reset.

When the TPU wins arbitration, it must respond to the CPU32 interrupt acknowledge cycle by placing an interrupt vector number on the data bus. The vector number is used to calculate displacement into the exception vector table. Vectors are formed by concatenating the 4-bit value of the CIBV field in TICR with the 4-bit number of the channel requesting interrupt service. Since the CIBV field has a reset value of 0x0, it must be assigned a value corresponding to the upper nibble of a block of 16 user-defined vector numbers before TPU interrupts are enabled. Otherwise, a TPU interrupt service request could cause the CPU32 to take one of the reserved vectors in the exception vector table.

5.3.8 Prescaler Control for TCR1

Timer count register 1 (TCR1) is clocked from the output of a prescaler. The following fields control TCR1:

- The PSCK and TCR1P fields in TPUMCR
- The DIV2 field in TPUMCR2
- The EPSCKE and EPSCK fields in TPUMCR3.

The rate at which TCR1 is incremented is determined as follows:

- The user selects either the standard prescaler (by clearing the enhanced prescaler enable bit, EPSCKE, in TPUMCR3) or the enhanced prescaler (by setting EPSCKE).
  - — If the standard prescaler is selected (EPSCKE = 0), the the PSCK bit determines whether the standard prescaler divides the system clock input by 32 (PSCK = 0) or four (PSCK = 1)
  - — If the enhanced prescaler is selected (EPSCKE = 1), the EPSCK bits select a value by which the system clock is divided. The lowest frequency for TCR1 clock is system clock divided by 64x8. The highest frequency for TCR1 clock is system clock divided by two (2 x 1). See **Table 5-1**.

Table 5-1 Enhanced TCR1 Prescaler Divide Values

EPSCK Value	Divide System Clock By
0x00	2
0x01	4
0x02	6
0x03	8
0x04, 0x05,...0x1d	10,12,...60
0x1e	62
0x1f	64

  - — The output of either the standard prescaler or the enhanced prescaler is then divided by 1, 2, 4, or 8, depending on the value of the TCR1P field in the TPUMCR.

Table 5-2 TCR1 Prescaler Values

TCR1P Value	Divide by
0b00	1
0b01	2
0b10	4
0b11	8

  - — If the DIV2 bit is one, the TCR1 counter increments at a rate of the internal clock divided by two. If DIV2 is zero, the TCR1 increment rate is defined by the output of the TCR1 prescaler (which, in turn, takes as input the output of either the standard or enhanced prescaler).

Figure 5-2 shows a diagram of the TCR1 prescaler control block.

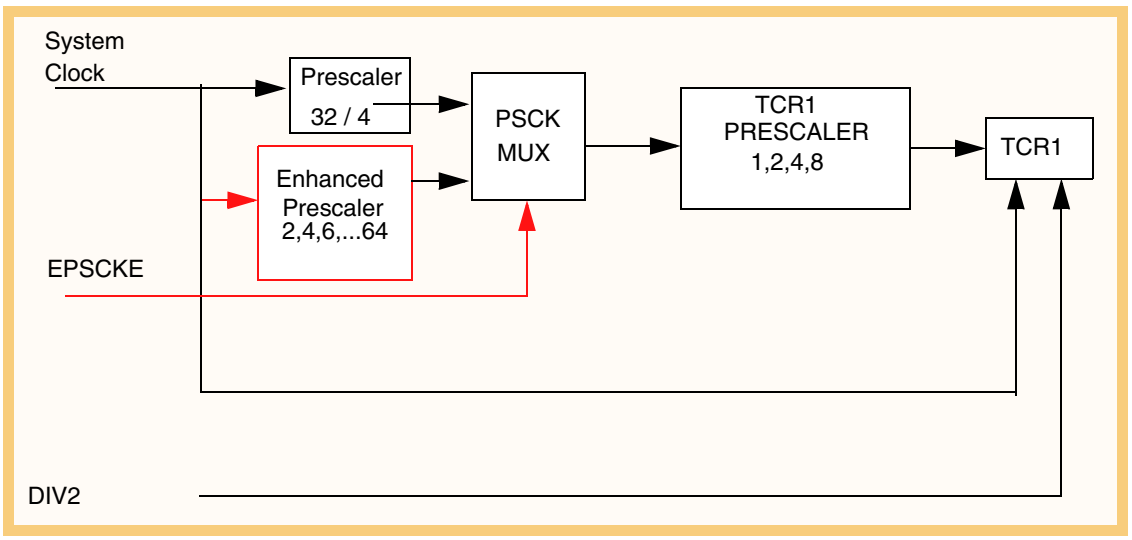


Figure 5-2 TCR1 Prescaler Control

5.3.9 Prescaler Control for TCR2

Timer count register 2 (TCR2), like TCR1, is clocked from the output of a prescaler. The T2CG (TCR2 clock/gate control) bit and the T2CSL (TCR2 counter clock edge) bit in TPUMCR determine T2CR2 pin functions. Refer to **Table 5-3**.

Table 5-3 TCR2 Counter Clock Source

T2CSL	T2CG	TCR2 Clock
0	0	Rise transition T2CLK
0	1	Gated system clock
1	0	Fall transition T2CLK
1	1	Rise & fall transition T2CLK

The function of the T2CG bit is shown in **Figure 5-3**.

When T2CG is set, the external T2CLK pin functions as a gate of the DIV8 clock (the TPU3 system clock divided by eight). In this case, when the external TCR2 pin is low, the DIV8 clock is blocked, preventing it from incrementing TCR2. When the external TCR2 pin is high, TCR2 is incremented at the frequency of the DIV8 clock. When T2CG is cleared, an external clock from the TCR2 pin, which has been synchronized and fed through a digital filter, increments TCR2. The duration between active edges on the T2CLK clock pin must be at least nine system clocks.

The TCR2PSCK2 bit in TPUMCR3 determines whether the clock source is divided by two before it is fed into the TCR2 prescaler. The TCR2 field in TPUMCR specifies the value of the prescaler: 1, 2, 4, or 8. Channels using TCR2 have the capability to

resolve down to the TPU3 system clock divided by eight. Figure 5-3 illustrates the TCR2 pre-divider and pre-scaler control.

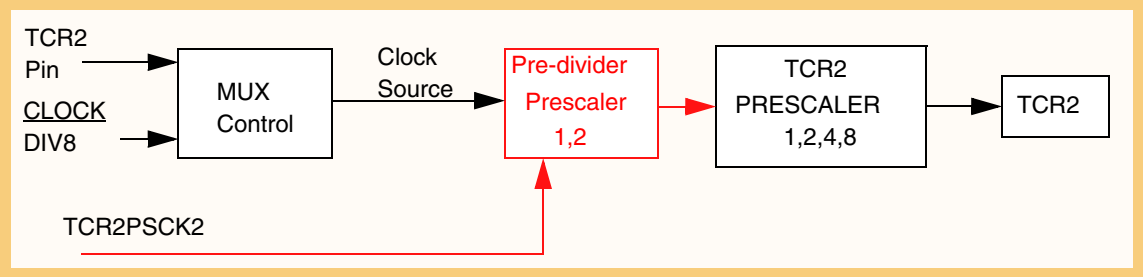


Figure 5-3 TCR2 Prescaler Control

Table 5-4 is a summary of prescaler output (assuming a divide-by-one value for the pre-divider prescaler.

Table 5-4 TCR2 Prescaler Control

TCR2 Prescaler	Divide By	Internal Clock Divided By	External Clock Divided By
00	1	8	1
01	2	16	2
10	4	32	4
11	8	64	8

5.4 Programming Model

The TPU3 memory map contains three groups of registers:

- System configuration registers
- Channel control and status registers
- Development support and test verification registers

All registers except the channel interrupt status register (CISR) must be read or written by means of half-word (16-bit) or word (32-bit) accesses. The address space of the TPU3 memory map occupies 512 bytes. Unused registers within the 512-byte address space return zeros when read.

Table 5-6 shows the TPU3 address map.

Table 5-5 TPU3_B Register Map

MSB 15 Address	LSB 0 Register
0xYF F800	TPU3 module configuration register (TPUMCR) See **Table 5-7** for bit descriptions.
0xYF F802	TPU3 test configuration register (TCR)
0xYF F804	Development support control register (DSCR) See **Table 5-8** for bit descriptions.
0xYF F806	Development support status register (DSSR) See **Table 5-9** for bit descriptions.
0xYF F808	TPU3 interrupt configuration register (TICR) See **Table 5-10** for bit descriptions.
0xYF F80A	Channel interrupt enable register (CIER) See **Table 5-11** for bit descriptions.
0xYF F80C	Channel function selection register 0 (CFSR0) See **Table 5-12** for bit descriptions.
0xYF F80E	Channel function selection register 1 (CFSR1) See **Table 5-12** for bit descriptions.
0xYF F810	Channel function selection register 2 (CFSR2) See **Table 5-12** for bit descriptions.
0xYF F812	Channel function selection register 3 (CFSR3) See **Table 5-12** for bit descriptions.
0xYF F814	Host sequence register 0 (HSQR0) See **Table 5-13** for bit descriptions.
0xYF F816	Host sequence register 1 (HSQR1) See **Table 5-13** for bit descriptions.
0xYF F818	Host service request register 0 (HSRR0) See **Table 5-14** for bit descriptions.
0xYF F81A	Host service request register 1 (HSRR1) See **Table 5-14** for bit descriptions.
0xYF F81C	Channel priority register 0 (CPR0) See **Table 5-15** for bit descriptions.
0xYF F81E	Channel priority register 1 (CPR1) See **Table 5-15** for bit descriptions.
0xYF F820	Channel interrupt status register (CISR) See **Table 5-17** for bit descriptions.
0xYF F822	Link register (LR)
0xYF F824	Service grant latch register (SGLR)
0xYF F826	Decoded channel number register (DCNR)
0xYF F828	TPU module configuration register 2 (TPUMCR2) See **Table 5-18** for bit descriptions.
0xYF F82A	TPU module configuration 3 (TPUMCR3) See **Table 5-21** for bit descriptions.
0xYF F82C	Internal scan data register (ISDR)
0xYF F82E	Internal scan control register (ISCR)
0xYF F900 – 0xYF F90F	Channel 0 parameter registers
0xYF F910 – 0xYF F91F	Channel 1 parameter registers

Table 5-5 TPU3_B Register Map (Continued)

MSB 15 Address	LSB 0 Register
0xYF F920 – 0xYF F92F	Channel 2 parameter registers
0xYF F930 – 0xYF F93F	Channel 3 parameter registers
0xYF F940 – 0xYF F94F	Channel 4 parameter registers
0xYF F950 – 0xYF F95F	Channel 5 parameter registers
0xYF F960 – 0xYF F96F	Channel 6 parameter registers
0xYF F970 – 0xYF F97F	Channel 7 parameter registers
0xYF F980 – 0xYF F98F	Channel 8 parameter registers
0xYF F990 – 0xYF F99F	Channel 9 parameter registers
0xYF F9A0 – 0xYF F9AF	Channel 10 parameter registers
0xYF F9B0 – 0xYF F9BF	Channel 11 parameter registers
0xYF F9C0 – 0xYF F9CF	Channel 12 parameter registers
0xYF F9D0 – 0xYF F9DF	Channel 13 parameter registers
0xYF F9E0 – 0xYF F9EF	Channel 14 parameter registers
0xYF F9F0 – 0xYF F9FF	Channel 15 parameter registers

Table 5-6 TPU3_A Register Map

MSB 15 Address	LSB 0 Register
0xYF FE00	TPU3 module configuration register (TPUMCR) See **Table 5-7** for bit descriptions.
0xYF FE02	TPU3 test configuration register (TCR)
0xYF FE04	Development support control register (DSCR) See **Table 5-8** for bit descriptions.
0xYF FE06	Development support status register (DSSR) See **Table 5-9** for bit descriptions.
0xYF FE08	TPU3 interrupt configuration register (TICR) See **Table 5-10** for bit descriptions.
0xYF FE0A	Channel interrupt enable register (CIER) See **Table 5-11** for bit descriptions.
0xYF FE0C	Channel function selection register 0 (CFSR0) See **Table 5-12** for bit descriptions.
0xYF FE0E	Channel function selection register 1 (CFSR1) See **Table 5-12** for bit descriptions.
0xYF FE10	Channel function selection register 2 (CFSR2) See **Table 5-12** for bit descriptions.
0xYF FE12	Channel function selection register 3 (CFSR3) See **Table 5-12** for bit descriptions.
0xYF FE14	Host sequence register 0 (HSQR0) See **Table 5-13** for bit descriptions.
0xYF FE16	Host sequence register 1 (HSQR1) See **Table 5-13** for bit descriptions.

Table 5-6 TPU3_A Register Map (Continued)

MSB 15 Address	LSB 0 Register
0xYF FE18	Host service request register 0 (HSRR0) See Table 5-14 for bit descriptions.
0xYF FE1A	Host service request register 1 (HSRR1) See Table 5-14 for bit descriptions.
0xYF FE1C	Channel priority register 0 (CPR0) See Table 5-15 for bit descriptions.
0xYF FE1E	Channel priority register 1 (CPR1) See Table 5-15 for bit descriptions.
0xYF FE20	Channel interrupt status register (CISR) See Table 5-17 for bit descriptions.
0xYF FE22	Link register (LR)
0xYF FE24	Service grant latch register (SGLR)
0xYF FE26	Decoded channel number register (DCNR)
0xYF FE28	TPU module configuration register 2 (TPUMCR2) See Table 5-18 for bit descriptions.
0xYF FE2A	TPU module configuration 3 (TPUMCR3) See Table 5-21 for bit descriptions.
0xYF FE2C	Internal scan data register (ISDR)
0xYF FE2E	Internal scan control register (ISCR)
0xYF FF00 – 0xYF FF0F	Channel 0 parameter registers
0xYF FF10 – 0xYF FF1F	Channel 1 parameter registers
0xYF FF20 – 0xYF FF2F	Channel 2 parameter registers
0xYF FF30 – 0xYF FF3F	Channel 3 parameter registers
0xYF FF40 – 0xYF FF4F	Channel 4 parameter registers
0xYF FF50 – 0xYF FF5F	Channel 5 parameter registers
0xYF FF60 – 0xYF FF6F	Channel 6 parameter registers
0xYF FF70 – 0xYF FF7F	Channel 7 parameter registers
0xYF FF80 – 0xYF FF8F	Channel 8 parameter registers
0xYF FF90 – 0xYF FF9F	Channel 9 parameter registers
0xYF FFA0 – 0xYF FFAF	Channel 10 parameter registers
0xYF FFB0 – 0xYF FFBF	Channel 11 parameter registers
0xYF FFC0 – 0xYF FFCF	Channel 12 parameter registers
0xYF FFD0 – 0xYF FFDF	Channel 13 parameter registers
0xYF FFE0 – 0xYF FFEF	Channel 14 parameter registers
0xYF FFF0 – 0xYF FFFF	Channel 15 parameter registers

5.4.1 TPU Module Configuration Register

TPUMCR — TPU Module Configuration Register

0xYF F800
0xYF FE00

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
STOP	TCR1P		TCR2P		EMU	T2CG	STF	SUPV	PSCK	TPU3	T2CSL	IARB[3:0]			
RESET:															
0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Table 5-7 TPUMCR Bit Settings

Bit(s)	Name	Description
15	STOP	Low-power stop mode enable. If the STOP bit in TPUMCR is set, the TPU3 shuts down its internal clocks, shutting down the internal microengine. TCR1 and TCR2 cease to increment and retain the last value before the stop condition was entered. The TPU3 asserts the stop flag (STF) in TPUMCR to indicate that it has stopped. 0 = Enable TPU3 clocks 1 = Disable TPU3 clocks
14:13	TCR1P	Timer count register 1 prescaler control. TCR1 is clocked from the output of a prescaler. The prescaler divides its input by 1, 2, 4, or 8. This is a write-once field unless the PWOD bit in TPUMCR3 is set. 00 = Divide by 1 01 = Divide by 2 10 = Divide by 4 11 = Divide by 8 Refer to **5.3.8 Prescaler Control for TCR1** for more information.
12:11	TCR2P	Timer count register 2 prescaler control. TCR2 is clocked from the output of a prescaler. The prescaler divides this input by 1, 2, 4, or 8. This is a write-once field unless the PWOD bit in TPUMCR3 is set. 00 = Divide by 1 01 = Divide by 2 10 = Divide by 4 11 = Divide by 8 Refer to **5.3.9 Prescaler Control for TCR2** for more information.
10	EMU	Emulation control. In emulation mode, the TPU3 executes microinstructions from DPTRAM exclusively. Access to the DPTRAM via the IMB3 is blocked, and the DPTRAM is dedicated for use by the TPU3. After reset, this bit can be written only once. 0 = TPU3 and DPTRAM operate normally 1 = TPU3 and DPTRAM operate in emulation mode
9	T2CG	TCR2 clock/gate control 0 = TCR2 pin used as clock source for TCR2 1 = TCR2 pin used as gate of DIV8 clock for TCR2 Refer to **5.3.9 Prescaler Control for TCR2** for more information.
8	STF	Stop flag. 0 = TPU3 is operating normally 1 = TPU3 is stopped (STOP bit has been set)
7	SUPV	Supervisor data space 0 = Assignable registers are accessible from user or supervisor privilege level 1 = Assignable registers are accessible from supervisor privilege level only
6	PSCK	Standard prescaler clock. Note that this bit has no effect if the extended prescaler is selected (EPSCKE = 1). 0 = $f_{SYS} \div 32$ is input to TCR1 prescaler, if standard prescaler is selected 1 = $f_{SYS} \div 4$ is input to TCR1 prescaler, if standard prescaler is selected

Table 5-7 TPUMCR Bit Settings (Continued)

Bit(s)	Name	Description
5	TPU3	TPU3 enable. The TPU3 enable bit provides compatibility with the TPU. If running TPU code on the TPU3, the microcode size should not be greater than 2 Kbytes and the TPU3 enable bit should be cleared to zero. The TPU3 enable bit is write-once after reset. The reset value is one, meaning that the TPU3 will operate in TPU3 mode. 0 = TPU mode; zero is the TPU reset value 1 = TPU3 mode; one is the TPU3 reset value **NOTE**: The programmer should not change this value unless necessary when developing custom TPU microcode.
4	T2CSL	TCR2 counter clock edge. This bit and the T2CG control bit determine the clock source for TCR2. Refer to **5.3.9 Prescaler Control for TCR2** for details.
3:0	IARB[3:0]	Interrupt Arbitration ID. The IARB field is used to arbitrate between simultaneous interrupt requests of the same priority. Each module that can generate interrupt requests must be assigned a unique, non-zero IARB field value.

5.4.2 Development Support Control Register

DSCR — Development Support Control Register

0xYF F804
0xYF FE04

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
HOT4	RESERVED				BLC	CLKS	FRZ		CCL	BP	BC	BH	BL	BM	BT
RESET:															
0					0	0	0	0	0	0	0	0	0	0	0

Table 5-8 DSCR Bit Settings

Bit(s)	Name	Description
15	HOT4	Hang on T4 0 = Exit wait on T4 state caused by assertion of HOT4 1 = Enter wait on T4 state
14:11	—	Reserved
10	BLC	Branch latch control 0 = Latch conditions into branch condition register before exiting halted state 1 = Do not latch conditions into branch condition register before exiting the halted state or during the time-slot transition period
9	CLKS	Stop clocks (to TCRs) 0 = Do not stop TCRs 1 = Stop TCRs during the halted state
8:7	FRZ	FREEZE assertion response. The FRZ bits specify the TPU microengine response to the IMB3 FREEZE signal 00 = Ignore freeze 01 = Reserved 10 = Freeze at end of current microcycle 11 = Freeze at next time-slot boundary
6	CCL	Channel conditions latch. CCL controls the latching of channel conditions (MRL and TDL) when the CHAN register is written. 0 = Only the pin state condition of the new channel is latched as a result of the write CHAN register microinstruction 1 = Pin state, MRL, and TDL conditions of the new channel are latched as a result of a write CHAN register microinstruction

Table 5-8 DSCR Bit Settings (Continued)

Bit(s)	Name	Description
5	BP	μPC breakpoint enable 0 = Breakpoint not enabled 1 = Break if μPC equals μPC breakpoint register
4	BC	Channel breakpoint enable 0 = Breakpoint not enabled 1 = Break if CHAN register equals channel breakpoint register at beginning of state or when CHAN is changed through microcode
3	BH	Host service breakpoint enable 0 = Breakpoint not enabled 1 = Break if host service latch is asserted at beginning of state
2	BL	Link service breakpoint enable 0 = Breakpoint not enabled 1 = Break if link service latch is asserted at beginning of state
1	BM	MRL breakpoint enable 0 = Breakpoint not enabled 1 = Break if MRL is asserted at beginning of state
0	BT	TDL breakpoint enable 0 = Breakpoint not enabled 1 = Break if TDL is asserted at beginning of state

5.4.3 Development Support Status Register

DSSR — Development Support Status Register

0xYF F806
0xYF FE06

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED								BKPT	PCBK	CHBK	SRBK	TPUF	RESERVED		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-9 DSSR Bit Settings

Bit(s)	Name	Description
15:8	—	Reserved
7	BKPT	Breakpoint asserted flag. If an internal breakpoint caused the TPU3 to enter the halted state, the TPU3 asserts the BKPT signal on the IMB and sets the BKPT flag. BKPT remains set until the TPU3 recognizes a breakpoint acknowledge cycle, or until the IMB FREEZE signal is asserted.
6	PCBK	μPC breakpoint flag. PCBK is asserted if a breakpoint occurs because of a μPC (microprogram counter) register match with the μPC breakpoint register. PCBK is negated when the BKPT flag is cleared.
5	CHBK	Channel register breakpoint flag. CHBK is asserted if a breakpoint occurs because of a CHAN register match with the CHAN register breakpoint register. CHBK is negated when the BKPT flag is cleared.
4	SRBK	Service request breakpoint flag. SRBK is asserted if a breakpoint occurs because of any of the service request latches being asserted along with their corresponding enable flag in the development support control register. SRBK is negated when the BKPT flag is cleared.

Table 5-9 DSSR Bit Settings (Continued)

Bit(s)	Name	Description
3	TPUF	TPU3 FREEZE flag. TPUF is set whenever the TPU3 is in a halted state as a result of FREEZE being asserted. This flag is automatically negated when the TPU3 exits the halted state because of FREEZE being negated.
2:0	—	Reserved

5.4.4 TPU3 Interrupt Configuration Register

TICR — TPU3 Interrupt Configuration Register **0xYF F808**
0xYF FE08

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED					CIRL			CIBV				RESERVED			
RESET:					0	0	0	0	0	0	0				

Table 5-10 TICR Bit Settings

Bit(s)	Name	Description
15:11	—	Reserved
10:8	CIRL	Channel interrupt request level. This three-bit field specifies the interrupt request level for all channels. T field is used in conjunction with the ILBS field to determine the request level of TPU3 interrupts.
7:4	CIBV	Channel interrupt base vector. The TPU is assigned 16 unique interrupt vector numbers, one vector number for each channel. The CIBV field specifies the most significant nibble of all 16 TPU channel interrupt vector numbers. The lower nibble of the TPU interrupt vector number is determined by the channel number on which the interrupt occurs.
3:0	—	Reserved.

5.4.5 Channel Interrupt Enable Register

The channel interrupt enable register (CIER) allows the CPU to enable or disable the ability of individual TPU3 channels to request interrupt service. Setting the appropriate bit in the register enables a channel to make an interrupt service request; clearing a bit disables the interrupt.

CIER — Channel Interrupt Enable Register **0xYF FE0A**
0xYF FE0A

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 15	CH 14	CH 13	CH 12	CH 11	CH 10	CH 9	CH 8	CH 7	CH 6	CH 5	CH 4	CH 3	CH 2	CH 1	CH 0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-11 CIER Bit Settings

Bit(s)	Name	Description
15:0	CH[15:0]	Channel interrupt enable/disable 0 = Channel interrupts disabled 1 = Channel interrupts enabled **NOTE:** The MSB (bit 0 in big-endian mode) represents CH15, and the LSB (bit 15 in big-endian mode) represents CH0.

5.4.6 Channel Function Select Registers

Encoded 4-bit fields within the channel function select registers specify one of 16 time functions to be executed on the corresponding channel. Encodings for predefined functions will be provided in a subsequent draft of this document.

CFSR0 — Channel Function Select Register 0 **0xYF F80C 0xYF FE0C**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 15				CH 14				CH 13				CH 12			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CFSR1 — Channel Function Select Register 1 **0xYF F80E 0xYF FE0E**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 11				CH 10				CH 9				CH 8			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CFSR2 — Channel Function Select Register 2 **0xYF F810 0xYF FE10**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 7				CH 6				CH 5				CH 4			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CFSR3 — Channel Function Select Register 3 **0xYF F812 0xYF FE12**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 3				CH 2				CH 1				CH 0			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-12 CFSRx Bit Settings

Name	Description
CH[15:0]	Encoded time function for each channel. Encoded four-bit fields in the channel function select registers specify one of 16 time functions to be executed on the corresponding channel.

5.4.7 Host Sequence Registers

The host sequence field selects the mode of operation for the time function selected on a given channel. The meaning of the host sequence bits depends on the time function specified. Meanings of host sequence bits and host service request bits for predefined time functions will be provided in a subsequent draft of this document.

HSQR0 — Host Sequence Register 0 **0xYF F814**
0xYF FE14

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 15		CH 14		CH 13		CH 12		CH 11		CH 10		CH 9		CH 8	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HSQR1 — Host Sequence Register 1 **0xYF F816**
0xYF FE16

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 7		CH 6		CH 5		CH 4		CH 3		CH 2		CH 1		CH 0	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-13 HSQRx Bit Settings

Name	Description
CH[15:0]	Encoded host sequence. The host sequence field selects the mode of operation for the time function selected on a given channel. The meaning of the host sequence bits depends on the time function specified.

5.4.8 Host Service Request Registers

The host service request field selects the type of host service request for the time function selected on a given channel. The meaning of the host service request bits is determined by time function microcode. Refer to Motorola's ***TPU Literature Package*, TPULITPAK/D,** for the available TPU documentation.

HSRR0 — Host Service Request Register 0

0xYF F818
0xYF FE18

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 15		CH 14		CH 13		CH 12		CH 11		CH 10		CH 9		CH 8	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HSRR1 — Host Service Request Register 1

0xYF F81A
0xYF FE1A

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 7		CH 6		CH 5		CH 4		CH 3		CH 2		CH 1		CH 0	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-14 HSSRx Bit Settings

Name	Description
CH[15:0]	Encoded type of host service. The host service request field selects the type of host service request for the time function selected on a given channel. The meaning of the host service request bits depends on the time function specified. A host service request field cleared to 0b00 signals the host that service is completed by the microengine on that channel. The host can request service on a channel by writing the corresponding host service request field to one of three non-zero states. The CPU must monitor the host service request register until the TPU3 clears the service request to 0b00 before any parameters are changed or a new service request is issued to the channel.

5.4.9 Channel Priority Registers

The channel priority registers (CPR1, CPR2) assign one of three priority levels to a channel or disable the channel.

CPR0 — Channel Priority Register 0

0xYF F81C
0xYF FE1C

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 15		CH 14		CH 13		CH 12		CH 11		CH 10		CH 9		CH 8	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CPR1 — Channel Priority Register 1

0xYF F81E
0xYF FE1E

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 7		CH 6		CH 5		CH 4		CH 3		CH 2		CH 1		CH 0	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-15 CPRx Bit Settings

Name	Description
CH[15:0]	Encoded channel priority levels. Table 5-16 indicates the number of time slots guaranteed for each channel priority encoding.

Table 5-16 Channel Priorities

CHx[1:0]	Service	Guaranteed Time Slots
00	Disabled	—
01	Low	1 out of 7
10	Middle	2 out of 7
11	High	4 out of 7

5.4.10 Channel Interrupt Status Register

The channel interrupt status register (CISR) contains one interrupt status flag per channel. Time functions specify via microcode when an interrupt flag is set. Setting a flag causes the TPU3 to make an interrupt service request if the corresponding CIER bit is set. To clear a status flag, read CISR, then write a zero to the appropriate bit. CISR is the only TPU3 register that can be accessed on a byte basis.

CISR — Channel Interrupt Status Register

0xYF F820
0xYF FE20

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 15	CH 14	CH 13	CH 12	CH 11	CH 10	CH 9	CH 8	CH 7	CH 6	CH 5	CH 4	CH 3	CH 2	CH 1	CH 0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-17 CISR Bit Settings

Bit(s)	Name	Description
15:0	CH[15:0]	Channel interrupt status 0 = Channel interrupt not asserted 1 = Channel interrupt asserted

5.4.11 TPU3 Module Configuration Register 2

TPUMCR2 — TPU Module Configuration Register 2

0xYF F828
0xYF FE28

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED							DIV2	SOFT RST	ETBANK		FPSCK			T2CF	DTPU
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-18 TPUMCR2 Bit Settings

Bit(s)	Name	Description
15:9	—	Reserved
8	DIV2	Divide by two control. When asserted, the DIV2 bit, along with the TCR1P bit and the PSCK bit in the TPUMCR, determines the rate of the TCR1 counter in the TPU3. If set, the TCR1 counter increments at a rate of two system clocks. If negated, TCR1 increments at the rate determined by control bits in the TCR1P and PSCK fields. 0 = TCR1 increments at rate determined by control bits in the TCR1P and PSCK fields of the TPUMCR register 1 = Causes TCR1 counter to increment at a rate of the system clock divided by two
7	SOFT RST	Soft reset. The TPU3 performs an internal reset when both the SOFT RST bit in the TPUMCR2 and the STOP bit in TPUMCR are set. The CPU must write zero to the SOFT RST bit to bring the TPU3 out of reset. The SOFT RST bit must be asserted for at least nine clocks. 0 = Normal operation 1 = Puts TPU3 in reset until bit is cleared **NOTE**: Do not attempt to access any other TPU3 registers when this bit is asserted. When this bit is asserted, it is the only accessible bit in the register.
6:5	ETBANK	Entry table bank select. This field determines the bank where the microcoded entry table is situated. After reset, this field is 0b00. This control bit field is write once after reset. ETBANK is used when the microcode contains entry tables not located in the default bank zero. To execute the ROM functions on this MCU, ETBANK[1:0] must be 00. Refer to **Table 5-19**. **NOTE**: This field should not be modified by the programmer unless necessary because of custom microcode.
4:2	FPSCK	Filter prescaler clock. The filter prescaler clock control bit field determines the ratio between system clock frequency and minimum detectable pulses. The reset value of these bits is zero, defining the filter clock as four system clocks. Refer to **Table 5-20**.
1	T2CF	T2CLK pin filter control. When asserted, the T2CLK input pin is filtered with the same filter clock that is supplied to the channels. This control bit is write once after reset. 0 = Uses fixed four-clock filter 1 = T2CLK input pin filtered with same filter clock that is supplied to the channels
0	DTPU	Disable TPU3 pins. When the disable TPU3 control pin is asserted, pin TP15 is configured as an input disable pin. When the TP15 pin value is zero, all TPU3 output pins are three-stated, regardless of the pins function. The input is not synchronized. This control bit is write once after reset. 0 = TP15 functions as normal TPU3 channel 1 = TP15 pin configured as output disable pin. When TP15 pin is low, all TPU3 output pins are in a high-impedance state, regardless of the pin function.

Table 5-19 Entry Table Bank Location

ETBANK	Bank
00	0
01	1
10	2
11	3

Table 5-20 System Clock Frequency/Minimum Guaranteed Detected Pulse

Filter Control	Divide By	20 MHz	33 MHz
000	4	200 ns	121 ns
001	8	400 ns	242 ns
010	16	800 ns	485 ns
011	32	1.6 µs	970 ns
100	64	3.2 µs	1.94 µs
101	128	6.4 µs	3.88 µs
110	256	12.8 µs	7.76 µs
111	512	25.6 µs	15.51 µs

5.4.12 TPU Module Configuration Register 3

TPUMCR3 — TPU Module Configuration Register 3

0xYF F82A
0xYF FE2A

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED							PWOD	TCR2PCK2	EP-SCKE	RESERVED	EPSCK				
RESET:					0	0	0	0	0	0	0				

Table 5-21 TPUMCR3 Bit Settings

Bit(s)	Name	Description
15:9	—	Reserved
8	PWOD	Prescaler write-once disable bit. The PWOD bit does not lock the EPSCK field and the EPSCKE bit. 0 = Prescaler fields in MCR are write-once 1 = Prescaler fields in MCR can be written anytime
7	TCR2PSCK2	TCR2 prescaler 2 0 = Prescaler clock source is divided by one 1 = Prescaler clock source is divided by two
6	EPSCKE	Enhanced pre-scaler enable 0 = Disable enhanced prescaler (use standard prescaler) 1 = Enable enhanced prescaler. System clock will be divided by the value in EPSCK field.
5	—	Reserved
4:0	EPSCK	Enhanced prescaler value that will be loaded into the enhanced prescaler counter. Prescaler value = (EPSCK + 1) x 2. Refer to **5.3.8 Prescaler Control for TCR1** for details.

5.4.13 TPU3 Parameter RAM

The channel parameter registers are organized as one hundred 16-bit words of RAM. Channels 0 to 15 have eight parameters. The parameter registers constitute a shared work space for communication between the CPU and the TPU3. The TPU3 can only access data in the parameter RAM, refer to Table 5-22.

Table 5-22 Parameter RAM Address Map[1]

Channel Number	Parameter							
	0	1	2	3	4	5	6	7
0	00	02	04	06	08	0A	0C	0E
1	10	12	14	16	18	1A	1C	1E
2	20	22	24	26	28	2A	2C	2E
3	30	32	34	36	38	3A	3C	3E
4	40	42	44	46	48	4A	4C	4E
5	50	52	54	56	58	5A	5C	5E
6	60	62	64	66	68	6A	6C	6E
7	70	72	74	76	78	7A	7C	7E
8	80	82	84	86	88	8A	8C	8E
9	90	92	94	96	98	9A	9C	9E
10	A0	A2	A4	A6	A8	AA	AC	AE
11	B0	B2	B4	B6	B8	BA	BC	BE
12	C0	C2	C4	C6	C8	CA	CC	CE
13	D0	D2	D4	D6	D8	DA	DC	DE
14	E0	E2	E4	E6	E8	EA	EC	EE
15	F0	F2	F4	F6	F8	FA	FC	FE

NOTES:
1. The base address of the parameter RAM is 0xYF FF00. The parameter RAM addresses should be added to the base address.

SECTION 6
DUAL-PORT TPU RAM (DPTRAM)

6.1 Introduction

The dual-port RAM module with TPU microcode storage support (DPTRAM) consists of a control register block and a 6-Kbyte array of static RAM, which can be used either as a microcode storage for TPU or as a general-purpose memory.

The DPTRAM module acts as a common memory on the IMB3 and allows the transfer of data to the two TPU3 modules. Therefore, the DPTRAM interface includes an IMB3 bus interface and two TPU3 interfaces. When the RAM is being used in microcode mode, the array is only accessible to the TPU3 via a separate local bus, and not via the IMB3.

The dual-port TPU3 RAM (DPTRAM) is intended to serve as fast, two-clock access, general-purpose RAM memory for the MCU. When used as general-purpose RAM, this module is accessed via the MCU's internal bus.

The DPTRAM module is powered by V_{DDL} in normal operation.

NOTE

The $V_{DDDPTRAM}$ pin does not have circuitry to allow for standby operation and should be connected to V_{DDL}.

The DPTRAM may also be used as the microcode control store for up to two TPU3 modules when placed in a special emulation mode. In this mode the DPTRAM array may only be accessed by either or both of the TPU3 units simultaneously via separate emulation buses, and not via the IMB3.

The DPTRAM contains a multiple input signature calculator (MISC) in order to provide RAM data corruption checking. The MISC reads each RAM address and generates a 32-bit data-dependent signature. This signature can then be checked by the host.

The DPTRAM supports soft defects detection (SDD).

6.2 Features

- Six Kbytes of static RAM
- Only accessible by the CPU if neither TPU3 is in emulation mode
- Low-power stop operation
  - — Entered by setting the STOP bit in the DPTMCR
  - — Applies only to IMB3 accesses and not to accesses from either TPU3 interface
- TPU microcode mode
  - — The DPTRAM array acts as a microcode storage for the TPU module. This provides a means executing TPU code out of DPTRAM instead of program-

ming it in the TPU ROM.

- Includes built in check logic which scans the array contents and calculates the RAM signature
- IMB3 bus interface
- Two TPU3 interface units
- Bytes, half-word or word accessible

6.3 DPTRAM Configuration and Block Diagram

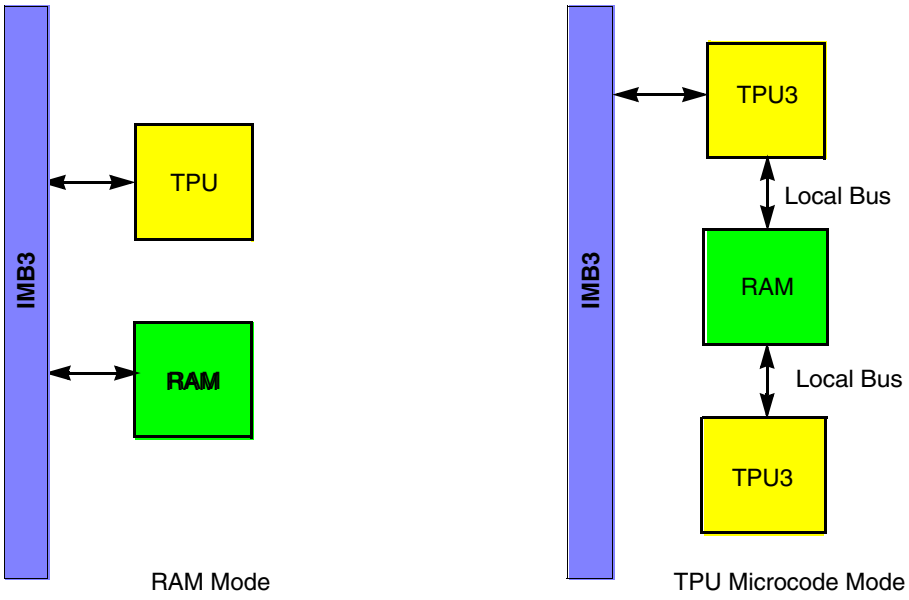


Figure 6-1 DPTRAM Configuration

6.4 Programming Model

The DPTRAM module consists of two separately addressable sections. The first is a set of memory-mapped control and status registers used for configuration (DPTMCR, RAMBAR, MISRH, MISRL, MISCNT) and testing (DPTTCR) of the DPTRAM array. The second section is the array itself.

All DPTRAM module control and status registers are located in supervisor data space. User reads or writes of these will result in a bus error.

When the TPU3 is using the RAM array for microcode control store, none of these control registers have any effect on the operation of the RAM array.

All addresses within the 64-byte control block will respond when accessed properly. Unimplemented addresses will return zeros for read accesses. Likewise, unimplemented bits within registers will return zero when read and will not be affected by write operations.

Table 6-1 shows the DPTRAM control and status registers. The addresses shown are offsets from the base address for the module. Refer to Figure 1-1 to locate the DPTRAM control block in the MC68377 address map. The DPTRAM array occupies a 6-Kbyte block.

Table 6-1 DPTRAM Register Map

R/$\overline{W}$ Access	Address	Register	Reset Value
Supv R/$\overline{W}$	0xYF F680	DPT RAM nodule configuration register (DPTMCR) See Table 6-2 for bit descriptions.	0x0100
Test	0xYF F682	Test configuration register (DPTTCR)	0x0000
Supv R/$\overline{W}$	0xYF F684	RAM base address register (DPTBAR) See Table 6-3 for bit descriptions.	0x0001
Supv read only	0xYF F686	Multiple Input Signature Register High (MISRH) See 6.4.4 for bit descriptions.	0x0000
Supv read only	0xYF F688	Multiple Input Signature Register Low (MISRL) See 6.4.4 for bit descriptions.	0x0000
Supv read only	0xYF F68A	Multiple Input Signature Counter (MISCNT) See 6.4.5 for bit descriptions.	Last memory address

6.4.1 DPTRAM Module Configuration Register (DPTMCR)

This register defines the basic configuration of the DPTRAM module. The DPTMCR contains bits to configure the DPT RAM module for stop operation and for proper access rights to the array. The register also contains the MISC control bits.

DPTMCR — DPT Module Configuration Register **0xYF F680**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
STOP	RESERVED				MISF	MIS-EN	RASP	RESERVED							
RESET:															
0					0	0	1	0	0	0	0	0	0	0	0

Table 6-2 DPTMCR Bit Settings

Bit(s)	Name	Description
15	STOP	Low power stop (sleep) mode 0 = DPTRAM clocks running 1 = DPTRAM clocks shut down Only the STOP bit in the DPTMCR may be accessed while the STOP bit is asserted. Accesses to other DPTRAM registers may result in unpredictable behavior. Note also that the STOP bit should be set and cleared independently of the other control bits in this register to guarantee proper operation. Changing the state of other bits while changing the state of the STOP bit may result in unpredictable behavior. Refer to **6.5.4 Stop Operation** for more information.
14:11	—	Reserved
10	MISF	Multiple input signature flag. MISF is readable at any time. This flag bit should be polled by the host to determine if the MISC has completed reading the RAM. If MISF is set, the host should read the MISRH and MISRL registers to obtain the RAM signature. 0 = First signature not ready 1 = MISC has read entire RAM. Signature is latched in MISRH and MISRL and is ready to be read.
9	MISEN	Multiple input signature enable. MISEN is readable and writable at any time. The MISC will only operate when this bit is set and the MC68377 is in TPU3 emulation mode. When enabled, the MISC will continuously cycle through the RAM addresses, reading each and adding the contents to the MISR. In order to save power, the MISC can be disabled by clearing the MISEN bit. 0 = MISC disabled 1 = MISC enabled
8	RASP	Ram area supervisor/user program/data. The RAM array may be placed in supervisor or unrestricted Space. When placed in supervisor space, (RASP = 1), only a supervisor may access the array. If a supervisor program is accessing the array, normal read/write operation will occur. If a user program is attempting to access the array, the access will be ignored and the address may be decoded externally. 0 = Both supervisor and user access to RAM allowed 1 = Supervisor access only to RAM allowed
7:0	—	Reserved

6.4.2 DPTRAM Test Register

DPTTCR — Test Register **0xYF F682**

DPTTCR is used only during factory testing of the MCU.

6.4.3 Ram Base Address Register (DPTBAR)

The DPTBAR register is used to specify the 16 MSBs of the starting DPT RAM array location in the memory map.

This register can be written only once after a reset and must be written after the DPRTAM is enabled (DPTMCR STOP = 0b0). This prevents runaway software from inadvertently re-mapping the array. Since the locking mechanism is triggered by the first write after reset, the base address of the array should be written in a single operation. Writing only one half of the register will prevent the other half from being written.

DPTBAR — RAM Array Base Address Register **0xYF F684**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
A23	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	RESERVED				RAMDS
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 6-3 DPTBAR Bit Settings

Bit(s)	Name	Description
15:5	A[8:18]	RAM array base address. These bits specify the 11 high-order bits (address lines ADDR[8:18] in little-endian notation) of the 24-bit base address of the RAM array. This allows the array to be placed on a 8-Kbyte boundary anywhere in the memory map. It is the users responsibility not to overlap the RAM array memory map with other modules on the chip.
4:1	—	Reserved. (Bits 11:12 represent A[12:11] in DPTRAM implementation that require them.)
0	RAMDS	RAM disabled. RAMDS is a read-only status bit. The RAM array is disabled after a master reset since the RAMBAR register may be incorrect. When the array is disabled, it will not respond to any addresses on the IMB3. Access to the RAM control register block is not affected when the array is disabled. RAMDS is cleared by the DPTRAM module when a base address is written to the array address field of RAMBAR. RAMDS = 0: RAM enabled RAMDS = 1: RAM disabled

6.4.4 MISR High (MISRH) and MISR Low (MISRL)

The MISRH and MISRL together contain the 32-bit RAM signature calculated by the MISC. These registers are read-only and should be read by the host when the MISF bit in the MCR is set.

NOTE

The naming of the D[31:0] bits represents little-endian bit encoding.

Exiting TPU3 emulation mode results in the reset of both MISRH and MISRL

MISRH — Multiple Input Signature Register High **0xYF F686**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
D31	D30	D29	D28	D27	D26	D25	D24	D23	D22	D21	D20	D19	D18	D17	D16
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MISRL — Multiple Input Signature Register Low **0xYF F688**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.4.5 MISC Counter (MISCNT)

The MISCNT contains the address of the current MISC memory access. This register is read-only.

NOTE

The naming of the A[31:0] bits represents little-endian bit encoding.

Exiting TPU3 emulation mode or clearing the MISEN bit in the DPTMCR results in the reset of this register.

MISCNT — MISC Counter **0xYF F68A**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	RESERVED		
RESET:															

Last Memory Address

6.5 Operation

The DPTRAM module has several modes of operation. The following sections describe DPTRAM operation in each of these modes.

6.5.1 Normal Operation

In normal operation, the DPTRAM is powered by V_{DDL} and may be accessed via the IMB3 by a bus master.

Read or write accesses of 8, 16, or 32 bits are supported. In normal operation, neither TPU3 can access the array.

6.5.2 Standby Operation

The DPTRAM on MC68377 does not support standby operation. The V_{DD} DPTRAM should always be powered up and down with V_{DDL}.

6.5.3 Reset Operation

When a synchronous reset occurs, a bus master is allowed to complete the current access. Thus a write bus cycle (byte or half word) that is in progress when a synchronous reset occurs will be completed without error. Once a write already in progress has been completed, further writes to the RAM array are inhibited.

NOTE

A word (32-bit) write will be completed coherently only if the reset occurs during the second (16-bit) write bus cycle. If reset occurs during the first write bus cycle, only the first half word will be written to the RAM array and the second write will not be allowed to occur. In this case, the word data contained in the DPTRAM will not be coherent. The first half word will contain the most significant half of the new word information and the second half word will contain the least significant half of the old word information.

If a reset is generated by an asynchronous reset such as the loss of clocks or software watchdog time-out, the contents of the RAM array are not guaranteed. (Refer to **6.5 Operation** for a description of MC68377 reset sources, operation, control, and status.)

Reset will also reconfigure some of the fields and bits in the DPTRAM control registers to their default reset state. See the description of the control registers to determine the effect of reset on these registers.

6.5.4 Stop Operation

Setting the STOP control bit in the DPTMCR causes the module to enter its lowest power-consuming state. The DPTMCR can still be written to allow the STOP control bit to be cleared.

In stop mode, the DPTRAM array cannot be read or written. All data in the array is retained. The BIU continues to operate to allow the CPU to access the STOP bit in the DPTMCR. The system clock remains stopped until the STOP bit is cleared or the DPTRAM module is reset.

The STOP bit is initialized to logical zero during reset. Only the STOP bit in the DPTMCR can be accessed while the STOP bit is asserted. Accesses to other DPTRAM registers may result in unpredictable behavior. Note also that the STOP bit should be set and cleared independently of the other control bits in this register to guarantee proper operation. Changing the state of other bits while changing the state of the STOP bit may result in unpredictable behavior.

6.5.5 Freeze Operation

The FREEZE line on the IMB3 has no effect on the DPTRAM module. When the freeze line is set, the DPTRAM module will operate in its current mode of operation. If the DPTRAM module is not disabled, (RAMDS = 0), it may be accessed via the IMB3. If the DPTRAM array is being used by the TPU in emulation mode, the DPTRAM can still accessed by the TPU microengine.

6.5.6 TPU3 Emulation Mode Operation

To emulate TPU3 time functions, the user stores the microinstructions required for all time functions, in the RAM array. This must be done with the DPTRAM in its normal operating mode and accessible from the IMB3. After the time functions are stored in

the array, the user places one or both of the TPU3 units in emulation mode. The RAM array is then controlled by the TPU3 units and disconnected from the IMB3.

To use the DPTRAM for microcode accesses, set the EMU bit in the corresponding TPU3 module configuration register. Through the auxiliary buses, the TPU3 units can access word instructions simultaneously at a rate of up to 40 MHz.

When the RAM array is being used by either or both of the TPU3 units, all accesses via the IMB3 are disabled. The control registers have no effect on the RAM array. Accesses to the array are ignored, allowing an external RAM to replace the function of the general-purpose RAM array.

The contents of the RAM are validated using a multiple input signature calculator (MISC). MISC reads of the RAM are performed only when MC68377 is in emulation mode and the MISC is enabled (MISEN = 1 in the DPTMCR).

Refer to **5.3.6 Emulation Support** for more information in TPU3 and DPTRAM operation in emulation mode.

6.6 Multiple Input Signature Calculator (MISC)

The integrity of the RAM data is ensured through the use of a MISC. The RAM data is read in reverse address order and a unique 32-bit signature is generated based on the output of these reads. MISC reads are performed when one of the TPU3 modules does not request back-to-back accesses to the RAM provided that the MISEN bit in the MC68377 MCR is set.

The MISC generates the DPTRAM signature based on the following polynomial:

$$G(x) = 1 + x + x^2 + x^{22} + x^{31}$$

After the entire RAM has been read and a signature has been calculated, the MISC sets the MISF bit in the MC68377 MCR. The host should poll this bit and enter a handling routine when the bit is found to be set.

The signature should be then read from the MISRH and MISRL registers and the host determines if it matches the predetermined signature.

The MISRH and MISRL registers are updated each time the MISC completes reading the entire RAM regardless of whether or not the previous signature has been read or not. This ensures that the host reads the most recently generated signature.

The MISC can be disabled by clearing the MISEN bit in the MC68377 MCR. Note that the reset state of the MC68377 MISEN is disabled.

SECTION 7
QUEUED SERIAL MODULE

The queued serial module (QSM) provides the microcontroller unit (MCU) with two serial communication interfaces divided into two submodules: the queued serial peripheral interface (QSPI) and the serial communications interface (SCI).

The QSPI is a full-duplex, synchronous serial interface for communicating with peripherals and other MCUs. It is enhanced by the addition of a queue for receive and transmit data.

The QSM's bus interface unit (BIU) is factory configured for each individual microcontroller. Refer to the ***Queued Serial Module Reference Manual*, (QSMRM/AD)**. The QSM's response to bus errors and interrupt request functionality are discussed in **7.5 Bus Error Support**.

The SCI is a full-duplex universal asynchronous receiver transmitter (UART) serial interface. The QSPI and SCI submodules operate independently.

This section provides a block diagram, memory map, pin description, and register descriptions of the QSM, with a breakdown of both the QSPI and SCI submodules. Operation of the QSPI submodule includes master mode and slave mode. For a detailed description refer to **7.3.5.1 Master Mode** and **7.3.5.4 Slave Mode**.

In addition, operation of the SCI submodule is divided into transmit and receive. A description of these operations is given in **7.4.3 Transmitter Operation** and **7.4.4 Receiver Operation**. To aid in grasping an understanding of the numerous bits and fields of the registers that appear throughout the text, a quick reference guide identifies all bit/field acronyms. (Refer to **Table 7-4**.)

7.1 Block Diagram

Figure 7-1 depicts the major components of the QSM, which consist of the global registers, logic control, and the QSPI and SCI submodules. Refer to **7.3 QSPI Submodule** and **7.4 SCI Submodule** for further definition of these components.

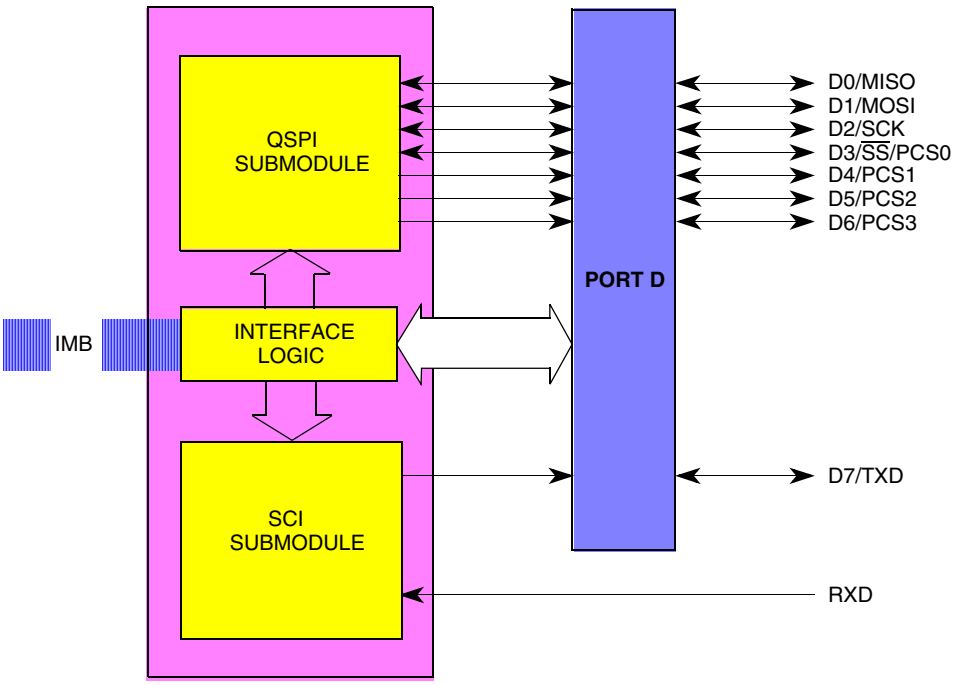
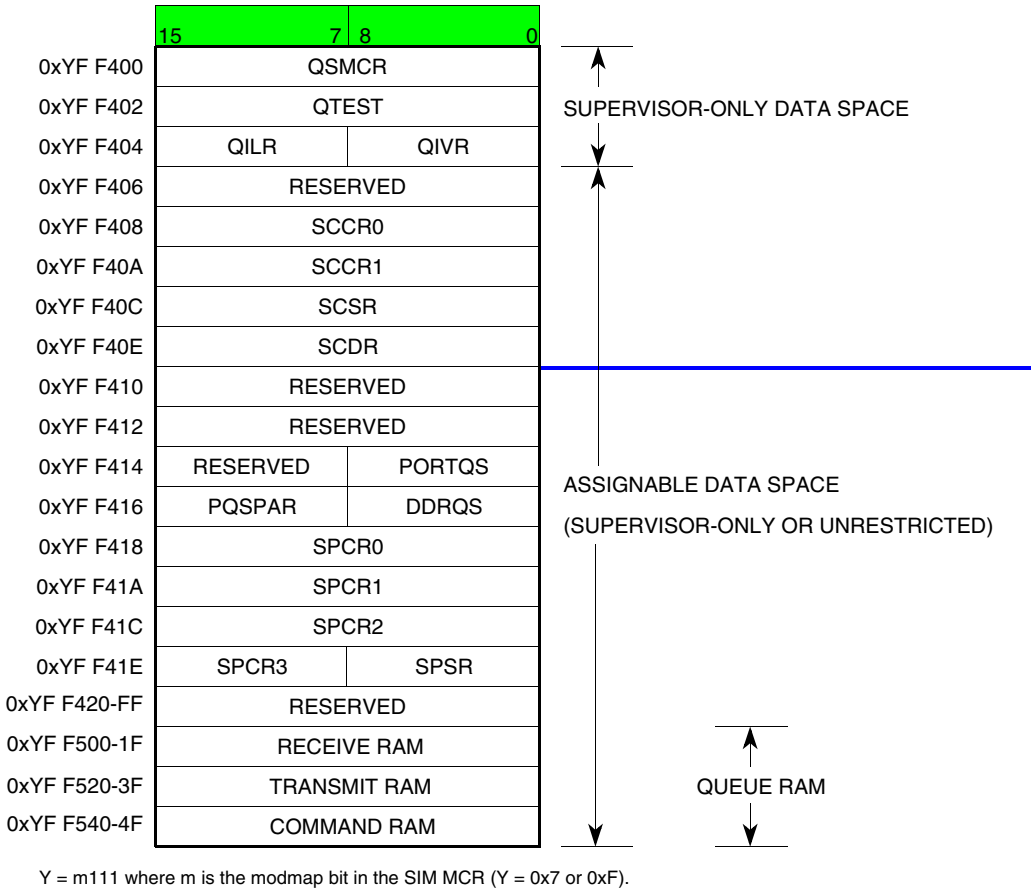


Figure 7-1 QSM Block Diagram

7.1.1 Memory Map

The QSM memory map is comprised of the global registers, the QSPI and SCI control and status registers, and the QSPI RAM as shown in **Figure 7-2**. For an accurate location of the QSM memory in the MCU memory map, refer to appropriate CPU manual. The QSM memory map may be divided into two segments: supervisor-only data space and assignable data space.



Y = m111 where m is the modmap bit in the SIM MCR (Y = 0x7 or 0xF).

Figure 7-2 QSM(B) Memory Map

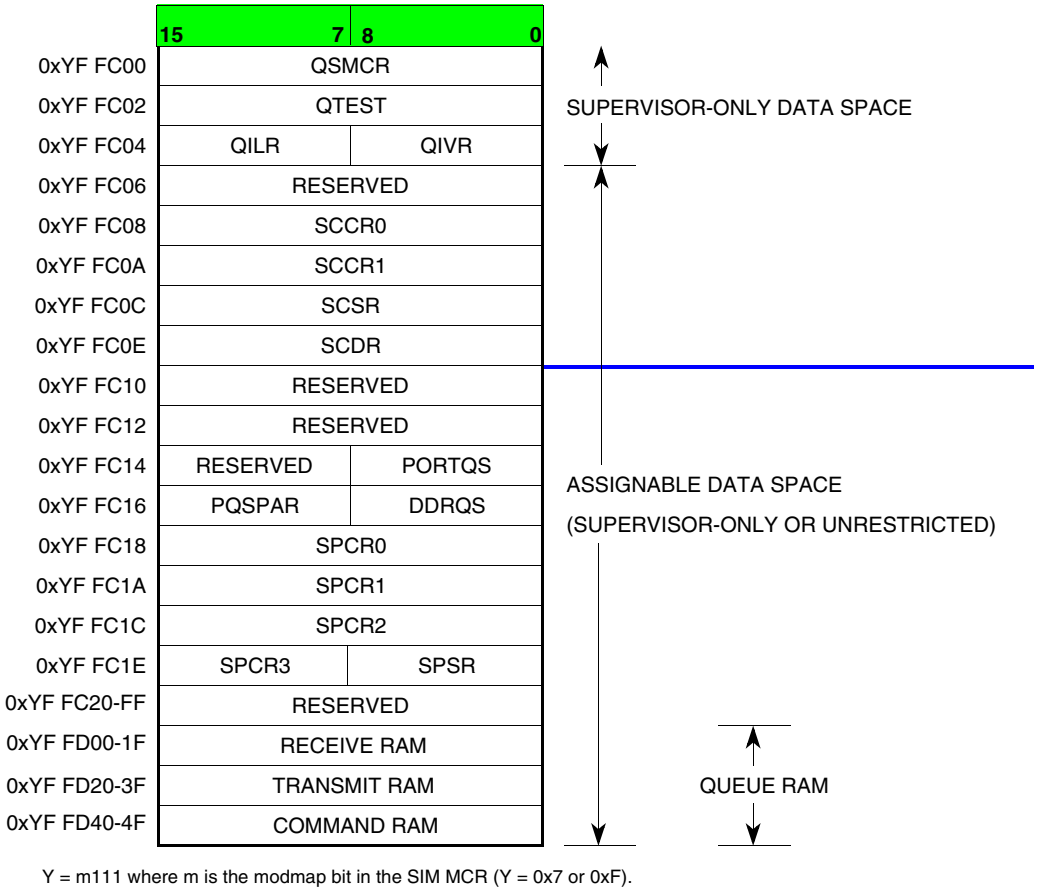


Figure 7-3 QSM_A Memory Map

The supervisor-only data space segment contains the QSM global registers. These registers define parameters needed by the QSM to integrate with the MCU. Access to these registers is permitted only when the CPU is operating in supervisor mode (CPU status register, S-bit = 1).

Assignable data space can be either restricted to supervisor-only access or unrestricted to both supervisor and user accesses. The supervisor (SUPV) bit in the QSM module configuration register (QSMCR) designates the assignable data space as either supervisor or unrestricted. If SUPV is set, then the space is designated as supervisor-only space. Access is then permitted only when the CPU is operating in supervisor mode. All attempts to read supervisor data spaces when not in supervisor mode (CPU status register, S-bit = 0) return a value of zero, and all attempts to write have no effect. If SUPV is clear, both user and supervisor accesses are permitted. To clear SUPV in the QSMCR, the CPU must be in supervisor mode (CPU status register, S-bit = 1). Refer to processing states in the appropriate CPU manual for more information on supervisor mode.

The QSM assignable data space segment contains the submodules, QSPI and SCI control/status registers, and the QSPI RAM. All registers and RAM may be accessed on byte, word, and long-word boundaries. The 80 bytes of static RAM are distinct from the QSM register set. All bytes not used by the QSPI may be used as general-purpose RAM. When operating, the QSPI submodule uses three non-contiguous blocks of the 80-byte RAM for receive, transmit, and control data. More information on the QSPI RAM can be found in **7.3.4.6 QSPI RAM**.

The contents of most locations in the memory map may be rewritten with the identical value to that location with one exception. (Refer to **7.3.4.3 QSPI Control Register 2 (SPCR2)**.) Writing a different value to certain control registers when a submodule using that register is enabled can cause unpredictable results. For predictable operation, disable the submodule in an orderly fashion before altering the registers.

7.1.2 SIGNAL DESCRIPTIONS

The QSM has nine external pins, as shown in **Figure 7-1**. Eight of the pins, if not in use for their submodule function, can be used as general-purpose I/O port pins. The ninth pin, RXD, is an input-only pin used exclusively by the SCI submodule.

The QSM pin control registers — DDRQS, QSM pin assignment register (PQSPAR, and QSM port data register (PORTQS) — affect pins being used as general-purpose I/O pins. The QSPI control register 0 (SPCR0) has one bit that affects seven pins employed as general-purpose output pins. Within this register the wired-OR mode (WOMQ) control bit determines whether MISO, MOSI, SCK, and PCS[3:0] function as open-drain output pins or as normal output pins, regardless of their use as general-purpose I/O pins or as QSPI output pins. Likewise, the SCI control register 1 (SCCR1) has one bit that affects the TXD pin when it is employed as a general-purpose output. In this register the wired-OR mode (WOMS) control bit determines whether TXD functions as an open-drain output pin or a normal output pin, regardless of this pin's use as a general-purpose output pin or as an SCI output pin. Refer to **7.2.3 QSM Pin Control Registers** for more information on these registers.

7.1.3 SCI Pins

There are two pins associated with the SCI, the RXD and TXD pins. The SCI pins and their functions are listed in **Table 7-1**.

7.1.3.1 RXD — Receive Data

This dedicated input signal furnishes serial data input to the SCI. The RXD pin cannot be used for general-purpose I/O.

7.1.3.2 TXD — Transmit Data

This signal is the serial data output from the SCI. TXD is available as a general-purpose I/O pin when the SCI transmitter is disabled. When used as a general-purpose I/O, TXD may be configured either as input or output as determined by the TXD bit in the QSM register DDRQS. The state of the TXD bit is ignored while the SCI is enabled. The TXD pin is enabled for SCI use by the transmitter enable bit (TE) in the SCI control

register 1 (SCCR1). Refer to **7.4.2.2 SCI Control Register 1 (SCCR1)** for more information.

Table 7-1 External Pin Inputs/Outputs to the SC

Pin Names	Mnemonics	Mode	Function
Receive data	RXD	Receiver disabled Receiver enabled	Not used Serial data input to SCI
Transmit data	TXD	Transmitter disabled Transmitter enabled	General-purpose I/O Serial data output from SCI

7.1.4 QSPI Pins

Seven pins are associated with the QSPI. When not needed for a QSPI application, they may be configured as general-purpose I/O pins. **Table 7-2** identifies the QSPI pins and their functions. QSM register DDRQS determines whether the pins are designated as input or output. The user must initialize DDRQS for the QSPI to function correctly.

7.1.4.1 PCS[3:0] — Peripheral Chip-Selects

These bidirectional signals provide QSPI peripheral chip-selects.

7.1.4.2 $\overline{SS}$ — Slave Select

Assertion of this bidirectional signal selects the QSPI when in slave mode. This is the same pin as PCS0.

7.1.4.3 SCK — QSPI Serial Clock

This bidirectional signal furnishes the clock from the QSPI in master mode or furnishes the clock to the QSPI in slave mode.

7.1.4.4 MISO — Master In Slave Out

This bidirectional signal furnishes serial data input to the QSPI in master mode, and serial data output from the QSPI in slave mode.

7.1.5 MOSI — Master Out Slave In

This bidirectional signal furnishes serial data output from the QSPI in master mode, and serial data input to the QSPI in slave mode.

Table 7-2 External Pin Inputs/Outputs to the QSPI

Pin Names	Mnemonics	Mode	Function
Master In Slave Out	**MISO**	**Master Slave**	Serial Data Input to QSPI Serial Data Output from QSPI
Master Out Slave In	**MOSI**	**Master Slave**	Serial Data Output from QSPI Serial Data Input to QSPI
Serial Clock	SCK[1]	Master Slave	Clock Output from QSPI Clock Input to QSPI
Peripheral Chip-Selects	PCS[3:1]	Master	Outputs Select Peripheral(s)
Peripheral Chip-Select[2] Slave Select[3]	PCS0/ $\overline{SS}$	Master Slave	Output Selects Peripheral(s) Input Selects the QSPI
Slave Select[4]	SS	Master	May Cause Mode Fault

NOTES:
1. All QSPI pins (except SCK) can be used as general-purpose I/O if they are not used by the QSPI while the QSPI is operating.
2. An output (PCS0) when the QSPI is in master mode.
3. An input ($\overline{SS}$) when the QSPI is in slave mode.
4. An input ($\overline{SS}$) when the QSPI is in master mode; useful in multimaster systems.

7.2 Configuration and Control

Registers of the QSM are divided into four categories: QSM global registers, QSM pin control registers, QSPI submodule registers, and SCI submodule registers. The QSPI and SCI registers are defined in **7.3.4 QSPI Programmer's Model and Registers** and **7.4.2 SCI Programmer's Model and Registers**, respectively. Writes to unimplemented bits have no meaning or effect, and reads from unimplemented bits always return a logic zero value.

The modmap bit of the system integration module (SIM) module configuration register (MCR) defines the most significant bit (ADDR23) of the address, shown in each register figure as Y (Y = 0x7 or 0xF). This bit, concatenated with the rest of the address given, forms the absolute address of each register.

Table 7-1 is a summary of the registers, bits, and reset states for the full QSM module.

As previously mentioned, **Table 7-2** is a quick reference guide to all the bits/fields of the QSM module. Along with the function, the register and register location of each bit/field are identified.

Table 7-3 QSM(B) Register Summary

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QSMCR 0xYF F400	STOP	FRZ1	FRZ0	0	0	0	0	0	SUPV	0	0	0	IARB			
RESET:	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
QTEST 0xYF F402	0	0	0	0	0	0	0	0	0	0	0	0	TSBD	SYNC	TQSM	TMM
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
QILR/QIVR 0xYF F404	0	0	ILQSPI			ILSCI			INTV							
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
0xYF F406	RESERVED															
SCCR0 0xYF F408	0	0	0	SCBR												
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
SCCR1 0xYF F40A	0	LOOPS	WOMS	ILT	PT	PE	M	WAKE	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SCSR 0xYF F40C	0	0	0	0	0	0	0	TDRE	TC	RDRF	RAF	IDLE	OR	NF	FE	PF
RESET:	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
SCDR 0xYF F40E	0	0	0	0	0	0	0	R8/T8	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0
RESET:	0	0	0	0	0	0	0	U	U	U	U	U	U	U	U	U
0xYF F410	RESERVED															
0xYF F412	RESERVED															
PORTQS 0xYF F414	0	0	0	0	0	0	0	0	DATA7 (TXD)	DATA6 (PCS3)	DATA5 (PCS2)	DATA4 (PCS1)	DATA3 (PCS0*)	DATA2 (SCK)	DATA1 (MOSI)	DATA0 (MISO)
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PQS-PAR/DDRQS 0xYF F416	0	PCS3	PCS2	PCS1	PCS0*	0	MOSI	MISO	TXD	PCS3	PCS2	PCS1	PCS0*	SCK	MOSI	MISO
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SPCR0 0xYF F418	MSTR	WOMQ	BITS				CPOL	CPHA	SPBR							
RESET:	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
SPCR1 0xYF F41A	SPE	DSCKL							DTL							
RESET:	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
SPCR2 0xYF F41C	SPIFIE	WREN	WRTO	0	ENDQP				0	0	0	0	NEWQP			
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SPCR3/SPSR 0xYF F41E	0	0	0	0	0	LOOPQ	HMIE	HALT	SPIF	MODF	HALTA	0	CPTQP			
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xYFFC20–0xYF F4FF	RESERVED															
RECEIVE RAM 0xYF FD00–0xYF F51F	QSPI RECEIVE DATA (16 WORDS)															
TRANSMIT RAM 0xYF F520–0xYF F53F	QSPI TRANSMIT DATA (16 WORDS)															
COMMAND RAM 0xYF F540–0xYF F54F	CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0*	CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0*

Table 7-4 QSM(A) Register Summary

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QSMCR 0xYF FC00	STOP	FRZ1	FRZ0	0	0	0	0	0	SUPV	0	0	0	IARB			
RESET:	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
QTEST 0xYF FC02	0	0	0	0	0	0	0	0	0	0	0	0	TSBD	SYNC	TQSM	TMM
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
QILR/QIVR 0xYF FC04	0	0	ILQSPI			ILSCI			INTV							
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
0xYF FC06	RESERVED															
SCCR0 0xYF FC08	0	0	0	SCBR												
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
SCCR1 0xYF FC0A	0	LOOPS	WOMS	ILT	PT	PE	M	WAKE	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SCSR 0xYF FC0C	0	0	0	0	0	0	0	TDRE	TC	RDRF	RAF	IDLE	OR	NF	FE	PF
RESET:	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
SCDR 0xYF FC0E	0	0	0	0	0	0	0	R8/T8	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0
RESET:	0	0	0	0	0	0	0	U	U	U	U	U	U	U	U	U
0xYF FC10	RESERVED															
0xYF FC12	RESERVED															
PORTQS 0xYF FC14	0	0	0	0	0	0	0	0	DATA7 (TXD)	DATA6 (PCS3)	DATA5 (PCS2)	DATA4 (PCS1)	DATA3 (PCS0*)	DATA2 (SCK)	DATA1 (MOSI)	DATA0 (MISO)
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PQS-PAR/DDRQS 0xYF FC16	0	PCS3	PCS2	PCS1	PCS0*	0	MOSI	MISO	TXD	PCS3	PCS2	PCS1	PCS0*	SCK	MOSI	MISO
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SPCR0 0xYF FC18	MSTR	WOMQ	BITS				CPOL	CPHA	SPBR							
RESET:	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
SPCR1 0xYF FC1A	SPE	DSCKL							DTL							
RESET:	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
SPCR2 0xYF FC1C	SPIFIE	WREN	WRTO	0	ENDQP				0	0	0	0	NEWQP			
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SPCR3/SPSR 0xYF FC1E	0	0	0	0	0	LOOPQ	HMIE	HALT	SPIF	MODF	HALTA	0	CPTQP			
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xYFFC20–0xYF FCFF	RESERVED															
RECEIVE RAM 0xYF FD00–0xYF FD1F	QSPI RECEIVE DATA (16 WORDS)															
TRANSMIT RAM 0xYF FD20–0xYF FD3F	QSPI TRANSMIT DATA (16 WORDS)															
COMMAND RAM 0xYF FD40–0xYF FD4F	CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0*	CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0*

Y = m111, where m is the modmap bit in the module configuration register for the SIM (Y = 0x7 or 0xF).

* The PCS0 bit listed above represents the dual-function PCS0/SS.

Table 7-5 Bit/Field Quick Reference Guide

Bit/Field Mnemonic	Function	Register	Register Location
SPBR	Serial Clock Baud Rate	SPCR0	QSPI
BITS	Bits Per Transfer	SPCR0	QSPI
BITSE	Bits Per Transfer Enable	QSPI RAM	QSPI
SCBR	Baud Rate	SCCR0	SCI
CONT	Continue	QSPI RAM	QSPI
CPHA	Clock Phase	SPCR0	QSPI
CPOL	Clock Polarity	SPCR0	QSPI
CPTQP	Completed Queue Pointer	SPSR	QSPI
DSCK	Peripheral Select Chip (PSC) to Serial Clock (SCK) Delay	QSPI RAM	QSPI
DSCKL	Delay before Serial Clock (SCK)	SPCR1	QSPI
DT	Delay after Transfer	QSPI RAM	QSPI
DTL	Length of Delay after Transfer	SPCR1	QSPI
ENDQP	Ending Queue Pointer	SPCR2	QSPI
FE	Framing Error Flag	SCSR	SCI
FRZ[1:0]	Freeze1–0	QSMCR	QSM
HALT	Halt	SPCR3	QSPI
HALTA	Halt Acknowledge Flag	SPSR	QSPI
HMIE	Halt Acknowledge Flag (HALTA) and Mode Fault Flag (MODF) Interrupt Enable	SPCR3	QSPI
IARB	Interrupt Arbitration Identification Number	QSMCR	QSM
IDLE	Idle Line Detected Flag	SCSR	SCI
ILIE	Idle Line Interrupt Enable	SCCR1	SCI
ILQSPI	Interrupt Level for QSPI	QILR	QSM
ILSCI	Interrupt Level of SCI	QILR	QSM
ILT	Idle Line Detect Type	SCCR1	SCI
INTV	Interrupt Vector	QIVR	QSM
LOOPS	SCI Loop Mode	SCCR1	SCI
LOOPQ	QSPI Loop Mode	SPCR3	QSPI
M	Mode Select (8/9 Bit)	SCCR1	SCI
MISO	Master In Slave Out	PQSPAR/ DDRQS/PORTQS	QSM
MODF	Mode Fault Flag	SPSR	QSPI
MOSI	Master Out Slave In	PQSPAR/ DDRQS/PORTQS	QSM
MSTR	Master/Slave Mode Select	SPCR0	QSPI
NEWQP	New Queue Pointer Value	SPCR2	QSPI
NF	Noise Error Flag	SCSR	SCI
OR	Overrun Error Flag	SCSR	SCI
PCS0/$\overline{SS}$	Peripheral Chip-Select/Slave Select	PQSPAR/ DDRQS/PORTQS	QSM

Table 7-5 Bit/Field Quick Reference Guide (Continued)

Bit/Field Mnemonic	Function	Register	Register Location
PCS[3:1]	Peripheral Chip-Selects	PQSPAR/ DDRQS/PORTQS	QSM
PE	Parity Enable	SCCR1	SCI
PF	Parity Error Flag	SCSR	SCI
PT	Parity Type	SCCR1	SCI
R[8:0]	Receive 8–0	SCDR	SCI
RAF	Receiver Active Flag	SCSR	SCI
RDRF	Receive Data Register Full Flag	SCSR	SCI
RE	Receiver Enable	SCCR1	SCI
RIE	Receiver Interrupt Enable	SCCR1	SCI
RWU	Receiver Wakeup	SCCR1	SCI
SBK	Send Break	SCCR1	SCI
SCK	Serial Clock	DDRQS/PORTQS	QSM
SPE	QSPI Enable	SPCR1	QSPI
SPIF	QSPI Finished Flag	SPSR	QSPI
SPIFIE	SPI Finished Interrupt Enable	SPCR2	QSPI
STOP	Stop	QSMCR	QSM
SUPV	Supervisor/Unrestricted	QSMCR	QSM
SYNC	SCI Baud Clock Sync Signal	QTEST	QSM
T[8:0]	Transmit 8–0	SCDR	SCI
TC	Transmit Complete Flag	SCSR	SCI
TCIE	Transmit Complete Interrupt Enable	SCCR1	SCI
TDRE	Transmit Data Register Empty Flag	SCSR	SCI
TE	Transmit Enable	SCCR1	SCI
TIE	Transmit Interrupt Enable	SCCR1	SCI
TMM	Test Memory Map	QTEST	QSM
TQSM	Test QSM Enable	QTEST	QSM
TSBD	SPI Test Scan Path Select	QTEST	QSM
TXD	Transmit Data	DDRQS/PORTQS	QSM
WAKE	Wakeup Type	SCCR1	SCI
WOMQ	Wired-OR Mode for QSPI Pins	SPCR0	QSPI
WOMS	Wired-OR Mode for SCI Pins	SCCR1	SCI
WREN	Wrap Enable	SPCR2	QSPI
WRTO	Wrap To Select	SPCR2	QSPI

7.2.1 Overall QSM Configuration Summary

After reset, the QSM remains in an idle state, requiring initialization of several registers before any serial operations may begin execution. The following registers, fields, and bits are fully described later in this section. A general sequence guide for initialization follows:

- QSMCR (refer to 7.2.2.1 QSM Configuration Register (QSMCR)

This register must be initialized to properly configure:

- Interrupt arbitration identification number used by the entire QSM module
- Supervisor/unrestricted bit (SUPV)
- FREEZE and/or STOP configuration; which should remain cleared to zero for normal operation.
- QIVR and QILR (refer to **7.2.2.2 QSM Interrupt Level Register (QILR)** and

These registers are written to choose the base vector number for the entire QSM module and individual interrupt levels for the QSPI and SCI submodules.

PORTQS and DDRQS (refer to **7.2.3.1 QSM Port Data Register (PORTQS)** and **7.2.3.3 QSM Data Direction Register (DDRQS)**

The pin control registers should be initialized in the order PORTQS and then DDRQS, thus establishing the default state and direction of the QSM pins.

For configuration of the QSPI submodule, initialize as follows:

- RAM (refer to **7.3.4.6 QSPI RAM**)
- PQSPAR (refer to **7.2.3.2 QSM Pin Assignment Register (PQSPAR)**

Assignment of appropriate pins to the QSPI must be made with this register.

- SPCR0 (refer to **7.3.4.1 QSPI Control Register 0 (SPCR0)**

The system designer must choose a transfer rate (baud) for operation in master mode, an appropriate clock phase, clock polarity, and the number of bits to be transferred in a serial operation. Master/slave mode select (MSTR) must be set to configure the QSPI for master mode or cleared to configure operation in slave mode. WOMQ should be set to enable or cleared to disable wired-OR mode operation.

- SPCR1 (refer to **7.3.4.2 QSPI Control Register 1 (SPCR1)**
  - — SPE must be set to enable the QSPI; this register should be written last.
  - — DTL allows the user to program a delay after any serial transfer, which is invoked by the DT bit for any serial transfer.
  - — DSCKL allows the user to set a delay before SCK (after PCS valid), which is invoked by the DSCK bit for any transfer.
- SPCR2 (refer to **7.3.4.3 QSPI Control Register 2 (SPCR2)**
  - — NEWQP and ENDQP, respectively, determine the beginning of a queue and the number of serial transfers (up to 16) to be considered a complete queue.
  - — WREN is set to enable queue wraparound, and WRTO helps determine the address used in wraparound mode.
  - — SPIFIE is set to enable interrupts when SPIF is asserted.
- SPCR3 (refer to **7.3.4.4 QSPI Control Register 3 (SPCR3)**

HALT may be used for program debug, and HMIE is set to enable CPU interrupts when HALTA or MODF is asserted; LOOPQ is set only to enable a feedback loop that can be used for self-test mode.

For configuration of the SCI submodule, initialize as follows:

- SCCR0 (refer to **7.4.2.1 SCI Control Register 0 (SCCR0)**

The system designer must choose a transfer rate (baud) for serial transfer operation.

- SCCR1 (refer to **7.4.2.2 SCI Control Register 1 (SCCR1)**
  - — The type of serial frame (8- or 9-bit) and the use of parity must be determined by M, PE, and PT.
  - — For receive operation, the system designer must consider use and type of wakeup (WAKE, RWU, ILT, ILIE). The receiver must be enabled (RE) and, usually, RIE should be set.
  - — For transmit operation, the transmitter must be enabled (TE) and, usually, TIE should be set. The use of wired-OR mode (WOMS) must also be decided. Once the transmitter is configured, data is not sent until TDRE and TC are cleared. To clear TDRE and TC, the SCSR read must be followed by a write to SCDR (either the lower byte or the entire word).

7.2.2 QSM Global Registers

The QSM global registers contain system parameters used by both the QSPI and the SCI submodules. These registers define parameters used by the QSM to interface with the CPU and other system modules. The four global registers are listed in **Table 7-6**.

Table 7-6 QSM Global Registers

Address	Name	Usage
0xYF FC00	QSMCR	QSM Configuration Register
0xYF FC02	QTEST	QSM Test Register
0xYF FC04	QILR	QSM Interrupt Level Register
0xYF FC05	QIVR	QSM Interrupt Vector Register

7.2.2.1 QSM Configuration Register (QSMCR)

QSMCR contains parameters for interfacing to the CPU and the intermodule bus (IMB). This register can be modified only when the CPU is in supervisor mode.

QSMCR(B) — QSM Configuration Register **0xYF F400**
QSMCR(A) **0xYF FC00**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP	FRZ1	RESERVED						SUPV	RESERVED			IARB			
RESET:															
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 7-7 QSMCR(A,B) Bit Settings

Bit(s)	Name	Description
15	STOP	Stop enable. STOP places the QSM into a low power state by disabling the system clock in most parts of the module. QSMCR is the only register guaranteed to be readable while STOP is asserted. The QSPI RAM is not readable; however, writes to RAM or any register are guaranteed valid while STOP is asserted. STOP may be negated by the CPU and by reset. The system software must stop each submodule before asserting STOP to avoid complications at restart and to avoid data corruption. The SCI submodule receiver and transmitter should be disabled, and the operation should be verified for completion before asserting STOP. The QSPI submodule should be stopped by asserting the HALT bit in SPCR3 and by asserting STOP after the HALTA flag is set. 0 = Normal QSM clock operation 1 = QSM clock operation stopped
14	FRZ1	Freeze1 bit. FRZ1 determines what action is taken by the QSM when the FREEZE signal of the IMB is asserted. FREEZE is asserted whenever the CPU enters the background mode. **WARNING:** Ignoring the FREEZE signal can cause unpredictable results in the background mode operation of the QSM, because the CPU is unable to service interrupt requests in this mode. If FRZ1 equals one when the FREEZE line is asserted, the QSM comes to an orderly halt on a transfer boundary as if HALT had been asserted. The output pins continue to drive their last state. Once the FREEZE signal is negated, the QSM module restarts automatically. 0 = Ignore the FREEZE signal on the IMB 1 = Halt the QSM (on transfer boundary)
13:8	—	Reserved
7	SUPV	Supervisor/unrestricted. All registers in the QSM are placed in supervisor-only space. For any access from within user mode, address acknowledge (AACK) is not returned and the bus cycle is transferred externally. SUPV defines the assignable QSM registers as either supervisor-only data space or unrestricted data space. 0 = Assigned registers are unrestricted (user access allowed) 1 = Assigned registers are restricted (only supervisor access allowed)
6:4	—	Reserved
3:0	IARB	Interrupt arbitration identification number. Each module that generates interrupts, including the QSM, must have an IARB field. The value in this field is used to arbitrate for the IMB when two or more modules generate simultaneous interrupts of the same priority level. No two modules can share the same IARB value. The reset value of the IARB field is 0x0, which prevents the QSM from arbitrating during an interrupt acknowledge cycle (IACK). The IARB field should be initialized by system software to a value between 0xF (highest priority) and 0x1 (lowest priority). Otherwise, any interrupts generated are identified by the CPU as spurious.

7.2.2.2 QSM Interrupt Level Register (QILR)

QILR(B) — QSM Interrupt Level Register **0xYF F404**
QILR(A) **0xYF FC04**

15	14	13	12	11	10	9	8	7	0
RESERVED		ILQSPI			ILSCI			QIVR*	
RESET:									
0	0	0	0	0	0	0	0		

* QIVR — QSM Interrupt Vector Register

Table 7-8 QILR(A,B) Bit Settings

Bit(s)	Name	Description
15:14	—	Reserved
13:11	ILQSPI	Interrupt level for QSPI. ILQSPI determines the priority level of all QSPI interrupts. This field should be programmed to a value between 0x0 (interrupts disabled) and 0x7 (highest priority). If both the QSPI and the SCI modules contain the same priority level (not equal to zero) and both modules simultaneously request interrupt servicing, the QSPI is given priority.
10:8	ILSCI	Interrupt level of SCI. ILSCI determines the priority level of all SCI interrupts. This field should be programmed to a value between 0x0 (interrupts disabled) and 0x7 (highest priority).
7:0	QIVR	QSM interrupt vector register. See Table 7-9.

7.2.2.3 QSM Interrupt Vector Register (QIVR)

At reset, QIVR is initialized to 0x0F, which corresponds to the uninitialized interrupt vector in the exception table. This vector is selected until QIVR is written. QIVR should be programmed to one of the user-defined vectors (0x40 – 0xFF) during initialization of the QSM.

QIVR(B) — QSM Interrupt Vector Register **0xYF F404**
QIVR(A) **0xYF FC04**

15	8	7	6	5	4	3	2	1	0
QILR*		INTV[7:0]							
RESET:									
		0	0	0	0	1	1	1	1

* QILR — QSM Interrupt Level Register

Table 7-9 QIVR(A,B) Bit Settings

Bit(s)	Name	Description
15:8	QILR	QSM interrupt level register. See Table 7-8.
7:0	INTV[7:0]	Interrupt level for QSPI. After initialization, QIVR determines which two vectors in the exception vector table are to be used for QSM interrupts. The QSPI and SCI submodules have separate interrupt vectors adjacent to each other. Both submodules use the same interrupt vector with the least significant bit (LSB) determined by the submodule causing the interrupt. The value of INTV0 used during an IACK cycle is supplied by the bus interface unit (BIU). During an IACK, INTV[7:1] are driven on the DATA[7:1] lines. The INTV0 drives line DATA0 with a zero for an SCI interrupt and with a one for a QSPI interrupt. Writes to INTV0 have no meaning or effect. Reads of INTV0 return a value of one. 0 = (INVT0 setting) SCI generates an interrupt 1 = (INVT0 setting) QSPI generates an interrupt

7.2.3 QSM Pin Control Registers

Table 7-6 identifies the three pin control registers of the QSM. The QSM determines the use of nine pins, eight of which form a parallel port on the MCU. Although these pins are used by the serial subsystems, any pin may alternately be assigned as general-purpose I/O on a pin by pin basis. For use of these pins as general-purpose I/O, they must not be assigned to the QSPI submodule in register PQSPAR. To avoid briefly driving incorrect data, the first byte to be output should be written before register DDRQS is configured for any output pins. DDRQS should then be written to determine the direction of data flow on the pins and to output the value contained in register PORTQS for all pins defined as outputs. Subsequent data for output is then written to PORTQS.

Table 7-10 QSM Pin Control Registers

Address	Name	Usage
0xYF FC15	PORTQS	QSM Port Data Register
0xYF FC16	PQSPAR	QSM Pin Assignment Register
0xYF FC17	DDRQS	QSM Data Direction Register

7.2.3.1 QSM Port Data Register (PORTQS)

PORTQS determines the actual input or output value of a QSM port pin if the pin is defined in PQSPAR as general-purpose I/O. All QSM port pins may be used as general-purpose I/O. Writes to this register affect the pins defined as outputs; reads of this register return the actual value of the pins.

PORTQS(A) — QSM Port Data Register **0xYF F414**
PORTQS(B) **0xYF FC14**

15 8	7	6	5	4	3	2	1	0
RESERVED	DATA7 (TXD)	DATA6 (PCS3)	DATA5 (PCS2)	DATA4 (PCS1)	DATA3 (PCS0/$\overline{SS}$)	DATA2 (SCK)	DATA1 (MOSI)	DATA0 (MISO)
RESET:	0	0	0	0	0	0	0	0

7.2.3.2 QSM Pin Assignment Register (PQSPAR)

PQSPAR determines which of the QSPI pins, with the exception of the SCK pin, are actually used by the QSPI submodule, and which pins are available for general-purpose I/O. Pins may be assigned to the QSPI or to function as general-purpose I/O on a pin-by-pin basis. QSPI pins designated by PQSPAR as general-purpose I/O are controlled only by DDRQS and PORTQS and the QSPI has no effect on these pins. PQSPAR does not affect the operation of the SCI submodule.

PQSPAR(B) — QSM Pin Assignment Register **0xYF F416**
PQSPAR(A) **0xYF FC16**

15	14	13	12	11	10	9	8	7	0
RESERVED	PCS3	PCS2	PCS1	PCS0/$\overline{SS}$	RESERVED	MOSI	MISO	DDRQS*	
RESET:									
0	0	0	0	0	0	0	0		

*QSM data direction register

Table 7-11 PQSPAR(A,B) Bit Settings

Bit(s)	Name	Description
15	—	Reserved
14:12	PCS[3:1]	Peripheral chip-selects.
11	PCS0/$\overline{SS}$	Peripheral chip-select 0/slave select.These bits determine whether the associated QSM port pins function as general-purpose I/O pins or are assigned to the QSPI submodule.
10	—	Reserved
9:8	MOSI/MOSO	Master out, slave in/master in, slave out. These bits determine whether the associated QSM port pin functions as a general-purpose I/O pin or is assigned to the QSPI submodule.
7:0	DDRQS	Data direction register. See See **Table 7-12**.

7.2.3.3 QSM Data Direction Register (DDRQS)

DDRQS sets each I/O pin, except for TXD, as an input or an output regardless of whether the QSPI submodule is enabled or disabled. All QSM pins are configured during reset as general-purpose inputs. (The QSPI and SCI are disabled.) The RXD pin remains an input pin dedicated to the SCI submodule and does not function as a general-purpose I/O pin.

DDRQS(B) — QSM Data Direction Register **0xYF F416**
DDRQS(A) **0xYF FC16**

15	8	7	6	5	4	3	2	1	0
PQSPAR*		TXD	PCS3	PCS2	PCS1	PCS0/$\overline{SS}$	SCK	MOSI	MISO
RESET:									
		0	0	0	0	0	0	0	0

* PQSPAR — QSM Pin Assignment Register

Table 7-12 DDRQS(A,B) Bit Settings

Bit(s)	Name	Description
15:8	PQSPAR	Pin assignment register. See Table 7-11.
7	TXD	Transmit data. This bit determines the direction of the TXD pin (input or output), only if the SCI transmitter is disabled. If the SCI transmitter is enabled, the TXD bit is ignored, and the TXD pin is forced to function as an output.
6:4	PCS[3:1]	Peripheral chip-selects.
3	PCS0/$\overline{SS}$	Peripheral chip-select 0/slave select.
2	SCK	Serial clock.
1	MOSI	Master out, slave in.
0	MOSO	Master in, slave out. Refer to 7.3.5.4 Slave Mode for additional details on this pin. All of the above bits determine the QSPI port pin operation to be input or output. 0 = Input 1 = Output

7.3 QSPI Submodule

The QSPI submodule communicates with external peripherals and other MCUs via synchronous serial bus. The QSPI is fully compatible with the serial peripheral interface (SPI) systems found on other Motorola devices such as the M68HC11 and M68HC05 families. It has all of the capabilities of the SPI system as well as several new features. The following paragraphs describe the features, block diagram, pin descriptions, programmer's model (memory map) inclusive of registers, and the master and slave operation of the QSPI.

7.3.1 Features

Standard SPI features are listed below, followed by a list of the additional features offered on the QSPI:

- Full-duplex, three-wire synchronous transfers
- Half-duplex, two-wire synchronous transfers
- Master or slave operation
- Programmable master bit rates
- Programmable clock polarity and phase
- End-of-transmission interrupt flag
- Master-master mode fault flag
- Easily interfaces to simple expansion parts (A/D converters, EEPROMS, display drivers, etc.)

QSPI-enhanced features are as follows:

- Programmable queue — up to 16 preprogrammed transfers
- Programmable peripheral chip-selects — four pins select up to 16 SPI chips
- Wraparound transfer mode — for autoscanning of serial A/D (or other) peripherals, with no CPU overhead
- Programmable transfer length — from 8–16 bits inclusive
- Programmable transfer delay — from 1 µs to 0.5 ms (at 16.78 MHz)
- Programmable queue pointer
- Continuous transfer mode — up to 256 bits

7.3.1.1 Programmable Queue

A programmable queue allows the QSPI to perform up to 16 serial transfers without CPU intervention. Each transfer corresponds to a queue entry containing all the information needed by the QSPI to independently complete one serial transfer. This unique feature greatly reduces CPU/QSPI interaction, resulting in increased CPU and system throughput.

7.3.1.2 Programmable Peripheral Chip-Selects

Four peripheral chip-select pins allow the QSPI to access up to 16 independent peripherals by decoding the four peripheral chip-select signals. Up to four independent peripherals can be selected by direct connection to a chip-select pin. The peripheral chip-selects simplify interfacing to two or more serial peripherals by providing dedicated peripheral chip-select signals, alleviating the need for CPU intervention.

7.3.2 Wraparound Transfer Mode

Wraparound transfer mode allows automatic, continuous re-execution of the preprogrammed queue entries. Newly transferred data replaces previously transferred data. Wraparound simplifies interfacing with A/D converters by automatically providing the CPU with the latest A/D conversions in the QSPI RAM. Consequently, serial peripherals appear as memory-mapped parallel devices to the CPU.

7.3.2.1 Programmable Transfer Length

The number of bits in a serial transfer is programmable from eight to 16 bits, inclusive. For example, ten bits could be used for communicating with an external 10-bit A/D converter. Likewise, a vacuum fluorescent display driver might require a 12-bit serial transfer. The programmable length simplifies interfacing to serial peripherals that require different data lengths.

7.3.2.2 Programmable Transfer Delay

An inter-transfer delay may be programmed from approximately one to 500 µs (using a 16.78-MHz system clock). For example, an A/D converter may require time between transfers to complete a new conversion. The default delay is one µs (17 clocks at 16.78-MHz). The programmable length of delay simplifies interfacing to serial peripherals that require delay time between data transfers.

7.3.2.3 Programmable Queue Pointer

The QSPI has a pointer that identifies the queue location containing the data for the next serial transfer. The CPU can switch from one task to another in the QSPI by writing to the queue pointer, changing the location in the queue that is to be transferred next. Otherwise, the pointer increments after each serial transfer. By segmenting the queue, multiple-task support can be provided by the QSPI.

7.3.2.4 Continuous Transfer Mode

The continuous transfer mode allows the user to send and receive an uninterrupted bit stream with a peripheral. A minimum of 8 bits and a maximum of 256 bits may be transferred in a single burst without CPU intervention. Longer transfers are possible; however, minimal CPU intervention is required to prevent loss of data. A one-µs pause (using a 16.78-MHz system clock) is inserted between each queue entry transfer.

7.3.3 Block Diagram

Figure 7-4 provides a block diagram of the QSPI submodule components.

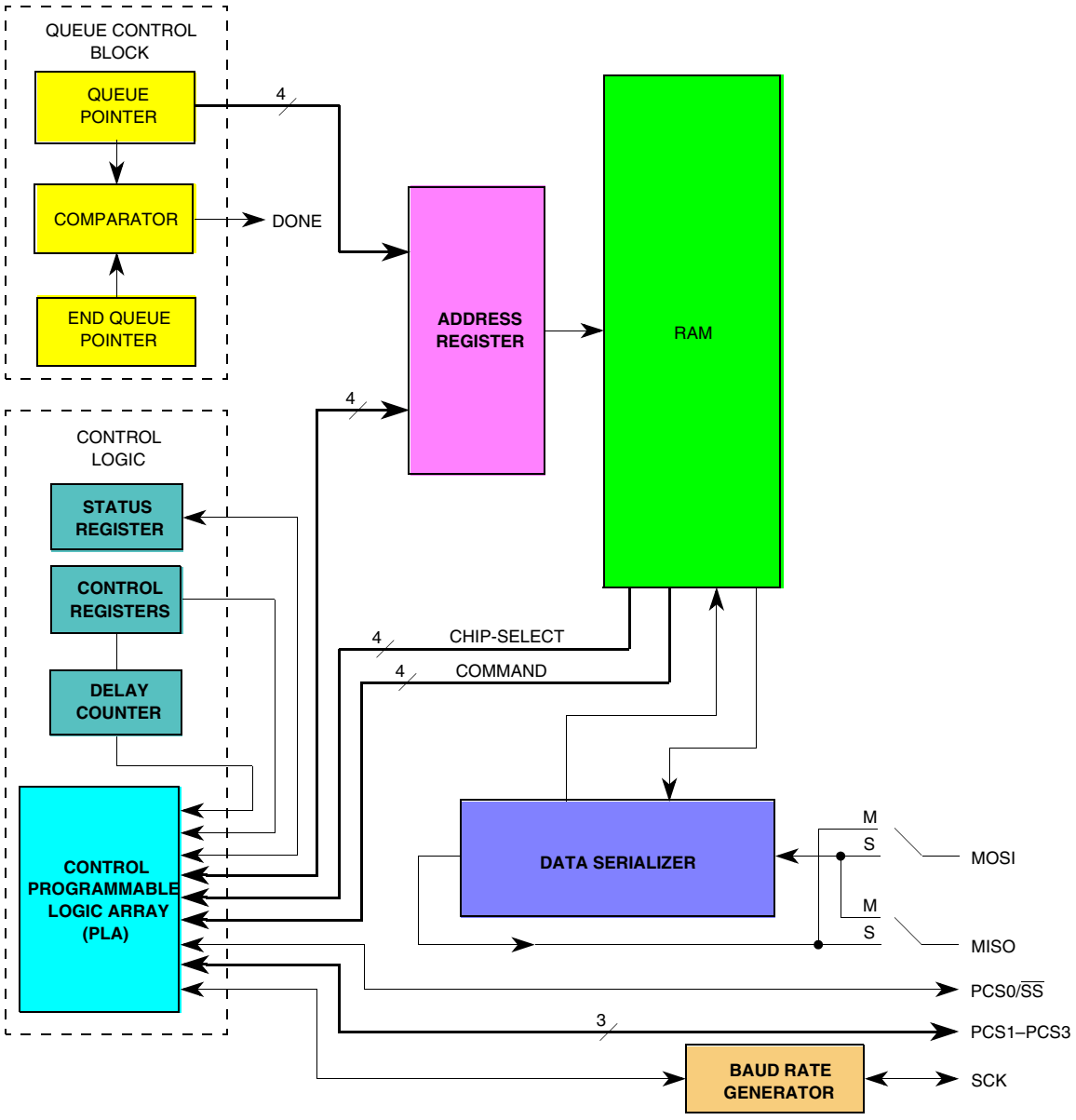


Figure 7-4 QSPI Submodule Diagram

7.3.4 QSPI Programmer's Model and Registers

The programmer's model (memory map) for the QSPI submodule consists of the QSM global and pin control registers (refer to **7.2.2 QSM Global Registers** and **7.2.3 QSM Pin Control Registers**), four QSPI control registers, one status register, and the 80-byte QSPI RAM. **Table 7-13** lists the registers and the QSPI RAM of the programmer's model. All of the registers and RAM can be read and written by the CPU. The four control registers must be initialized in proper order before the QSPI is enabled to ensure defined operation. Only the control registers must adhere to the order of sequence prescribed in **7.2.1 Overall QSM Configuration Summary**. Write register SPCR1 last when setting up the QSPI, as this register contains the QSPI enable bit (SPE). Asserting this bit starts the QSPI. QSPI control registers are reset to a defined state and may then be changed by the CPU. Reset values are shown below each register.

Table 7-13 QSPI Registers

Address	Name	Usage
0xYF F418, 9 (B) 0xYF FC18, 9 (A)	SPCR0	QSPI Control Register 0
0xYF F41A, B (B) 0xYF FC1A, B (A)	SPCR1	QSPI Control Register 1
0xYF F41C, D (B) 0xYF FC1C, D (A)	SPCR2	QSPI Control Register 2
0xYF F41E (B) 0xYF FC1E (A)	SPCR3	QSPI Control Register 3
0xYF F41F (B) 0xYF FC1F (A)	SPSR	QSPI Status Register
0xYF F500–1F (B) 0xYF FD00–1F (A)	RAM	QSPI Receive Data (16 Words)
0xYF F520–3F (B) 0xYF FD20–3F (A)	RAM	QSPI Transmit Data (16 Words)
0xYF F540–4F (B) 0xYF FD40–4F (A)	RAM	QSPI Command Control (8 Words)

In general, rewriting the same value into a control register does not affect the QSPI operation with the exception of NEWQP (bits [3:0]) in SPCR2. Rewriting the same value to these bits causes the RAM queue pointer to restart execution at the designated location.

If control bits are to be changed, the CPU should halt the QSPI first. With the exception of SPCR2, writing a different value into a control register while the QSPI is enabled may disrupt operation. SPCR2 is buffered, preventing any disruption of the current serial transfer. After completion of the current serial transfer, the new SPCR2 values become effective.

7.3.4.1 QSPI Control Register 0 (SPCR0)

SPCR0 contains parameters for configuring the QSPI before it is enabled. Although the CPU can read and write this register, the QSM has read-only access.

SPCR0(B) — QSPI Control Register 0 **0xYF F418**
SPCR0(A) **0xYF FC18**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSTR	WOMQ	BITS				CPOL	CPHA	SPBR							
RESET:															
0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0

Table 7-14 SPCR0(A,B) Bit Descriptions

Bit(s)	Name	Description
15	MSTR	Master/slave mode select. MSTR configures the QSPI for either master or slave mode operation. This bit is cleared on reset and may only be written by the CPU, not the QSM. 0 = QSPI is a slave device, and only responds to externally generated serial transfers. 1 = QSPI is system master and can initiate transmission to external SPI devices.
14	WOMQ	Wired-OR mode for QSPI pins. WOMQ allows the QSPI pins to be wire-ORed, regardless of whether they are used as general-purpose outputs or as QSPI outputs. WOMQ affects the QSPI pins whether the QSPI is enabled or disabled. This bit does not affect the SCI submodule transmit (TXD) pin, which has its own WOMS bit in an SCI control register. 0 = Output pins have normal outputs instead of open-drain outputs. 1 = All QSPI port pins designated as output by DDRQS function as open drain outputs and can be wire-ORed to other external lines.
13:10	BITS	Bits per transfer. In master mode, BITS determines the number of data bits transferred for each serial transfer in the queue that has the command control bit (BITSE of the QSPI RAM) equal to one. If BITSE equals zero for a command, eight bits are transferred for that command regardless of the value in BITS. Data transfers from eight to 16 bits are supported. Illegal (reserved) values all default to 8 bits. BITSE is not used in slave mode. All transfers are of the length specified by BITS. 0000 = 16 bits 0001 = Reserved 0010 = Reserved 0011 = Reserved 0100 = Reserved 0101 = Reserved 0110 = Reserved 0111 = Reserved 1000 = 8 bits 1001 = 9 bits 1010 = 10 bits 1011 = 11 bits 1100 = 12 bits 1101 = 13 bits 1110 = 14 bits 1111 = 15 bits
9	CPOL	Clock polarity. CPOL is used to determine the inactive state value of the serial clock (SCK). CPOL is used in conjunction with CPHA to produce the desired clock-data relationship between master and slave device(s). QSPI clock/data timing relationships are specified in individual microcontroller user's manuals. 0 = The inactive state value of SCK is low 1 = The inactive state value of SCK is high

Table 7-14 SPCR0(A,B) Bit Descriptions (Continued)

Bit(s)	Name	Description
8	CPHA	Clock phase. CPHA determines which edge of SCK causes data to change and which edge of SCK causes data to be captured. CPHA is used in conjunction with CPOL to produce the desired clock-data relationship between master and slave device(s). CPHA is set at reset. 0 = Data is captured on the leading edge of SCK and changed on the following edge of SCK. 1 = Data is changed on the leading edge of SCK and captured on the following edge of SCK.
7:0	SPBR	Serial clock baud rate. The QSPI internally generates the baud rate for SCK, the frequency of which is programmable by the user. The clock signal is derived from the MCU system clock using a modulus counter. At reset, BAUD is initialized to a 2.1-MHz SCK frequency (16.78-MHz system clock). The user programs a baud rate for SCK by writing a baud value from two to 255. The following equation determines the SCK baud rate: SCK Baud Rate = System Clock / (2 * SPBR) or SPBR = System Clock / (2 * SCK Baud Rate Desired) where SPBR equals 2, 3, 4,..., 255. Programming SPBR with the values zero or one disables the QSPI baud rate generator. SCK is disabled and assumes its inactive state value. No serial transfers occur. SPBR has 254 active values. See Table 7-14 for examples of serial clock frequencies.

Table 7-15 Examples of SCK Frequencies

System Clock Frequency	Required Division Ratio	Value of SPBR	Actual SCK Frequency
16.78 MHz	4	2	4.19 MHz
	8	4	2.10 MHz
	16	8	1.05 MHz
	34	17	493 kHz
	168	84	100 kHz
	510	255	33 kHz

7.3.4.2 QSPI Control Register 1 (SPCR1)

SPCR1 contains parameters for configuring the QSPI before it is enabled. Although the CPU can read and write this register, the QSM has read access only, except for SPE. This bit is automatically cleared by the QSPI after completing all serial transfers or when a mode fault occurs.

SPCR1(B) — QSPI Control Register 1 **0xYF F41A**
SPCR1(A) **0xYF FC1A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPE	DSCKL							DTL							
RESET:															
0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0

Table 7-16 SPCR1_A and SPCR1_B Bit Descriptions

Bit(s)	Name	Description
15	SPE	Master/slave mode select. This bit enables or disables the QSPI submodule. Setting SPE causes the QSPI to begin operation. If the QSPI is a master, setting SPE causes the QSPI to begin initiating serial transfers. If the QSPI is a slave, the QSPI begins monitoring the PCS0/$\overline{SS}$ pin to respond to the external initiation of a serial transfer. When the QSPI is disabled, the CPU may use the QSPI RAM. When the QSPI is enabled, both the QSPI and the CPU have access to the QSPI RAM. The CPU has both read and write access capability to all 80 bytes of the QSPI RAM. The QSPI can read only the transmit data segment and the command control segment, and can write only the receive data segment of the QSPI RAM. The QSPI turns itself off automatically when it is finished by clearing SPE. An error condition called mode fault (MODF) also clears SPE. This error occurs when PCS0/$\overline{SS}$ is configured for input, the QSPI is a system master (MSTR = 1), and PCS0/$\overline{SS}$ is driven low externally. To stop the QSPI, assert the HALT bit in SPCR3, then wait until the HALTA bit in SPSR is set. SPE may then be safely cleared to zero, providing an orderly method of quickly shutting down the QSPI after the current serial transfer is completed. The CPU can immediately disable the QSPI by just clearing SPE; however, loss of data from a current serial transfer may result and confuse an external SPI device. 0 = The QSPI is disabled, and the seven QSPI pins can be used as general-purpose I/O pins, regardless of the values in PQSPAR. 1 = The QSPI is enabled and the pins allocated by QSM register PQSPAR are controlled by the QSPI.
14:8	DSCKL	Delay before SCK. This bit determines the length of time the QSPI delays from peripheral chip-select (PCS) valid to SCK transition for serial transfers in which the command control bit, DSCK of the QSPI RAM, equals one. PCS may be any of the four peripheral chip-select pins. The following equation determines the actual delay before SCK: PCS to SCK Delay = [DSCKL/System Clock Frequency] where DSCKL equals {1,2,3,... 127}. A zero value for DSCKL causes a delay of 128 / system clocks, which equals 7.6 µs for a 16.78-MHz system clock. Because of design limits, a DSCKL value of one defaults to the same timing as a value of two. If a queue entry's DSCK equals zero, then DSCKL is not used. Instead, the PCS valid-to-SCK transition is one-half SCK period.
7:0	DTL	Length of delay after transfer. These bits determine the length of time that the QSPI delays after each serial transfer in which the command control bit, DT of the QSPI RAM, equals one. The following equation is used to calculate the delay: Delay after transfer = [(32 * DTL) / system clock frequency] where DTL equals {1, 2, 3,... 255}. A zero value for DTL causes a delay-after-transfer value of (32 * 256) / system clock, which equals 488.5 µs with a 16.78-MHz system clock. If DT equals zero, a standard delay is inserted. Standard delay-after-transfer = [17 / system clock] = 1 µs with a 16.78-MHz system clock Delay after transfer can be used to ensure that the deselect time requirement (for peripherals having such a requirement) is met. Some peripherals must be deselected for a minimum period of time between consecutive serial transfers. A delay after transfer can be inserted between consecutive transfers to a given peripheral to ensure that its minimum deselect time requirement is met or to allow serial A/D converters to complete conversion before the next transfer is made.

7.3.4.3 QSPI Control Register 2 (SPCR2)

SPCR2 contains parameters for configuring the QSPI. Although the CPU can read and write this register, the QSM has read access only. Writes to this register are buffered. A write to SPCR2 that changes any of the bit values (while the QSPI is operating) is ineffective on the current serial transfer, but becomes effective on the next serial trans-

fer. Reads of SPCR2 return the actual current value of the register, not the buffer. Refer to **7.3.5 Operating Modes and Flowcharts** for a detailed description of this register.

SPCR2_B — QSPI Control Register 2 **0xYF F41C**
SPCR2_A **0xYF FC1C**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPIFIE	WREN	WRTO	RESERVED	ENDQP				RESERVED				NEWQP			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 7-17 SPCR2_B and SPCR2_A Bit Descriptions

Bit(s)	Name	Description
15	SPIFE	SPI finished interrupt enable. SPIFIE enables the QSPI to generate a CPU interrupt upon assertion of the status flag SPIF. Because it is buffered, the value written to SPIFIE applies only upon completion of the queue (the transfer of the entry indicated by ENDQP). Thus, if a single sequence of queue entries is to be transferred (i.e., no WRAP), then SPIFIE should be set to the desired state before the first transfer. If a subqueue (see bit NEWQP) is to be used, the same CPU write that causes a branch to the subqueue may enable or disable the SPIF interrupt for the subqueue. The primary queue retains its own selected interrupt mode, either enabled or disabled. The SPIF interrupt must be cleared by clearing SPIF. Later interrupts may then be prevented by clearing SPIFIE to zero. The QSPI has three possible interrupt sources, but only one interrupt vector. These sources are SPIF, MODF, and HALTA. When the CPU responds to a QSPI interrupt, the user must ascertain the exact interrupt cause by reading register SPSR. Any interrupt that was set may then be cleared by writing to SPSR with a zero in the bit position corresponding to the exact interrupt source. Clearing SPIFIE does not immediately clear an interrupt already caused by SPIF.0 = The QSPI is disabled, and the seven QSPI pins can be used as general-purpose I/O pins, regardless of the values in PQSPAR. 0 = QSPI interrupts disabled 1 = QSPI interrupts enabled
14	WREN	Wrap enable. WREN enables or disables wraparound mode. If enabled, the QSPI executes commands in the queue through the command contained in ENDQP. Execution continues at either address 0x0 or at the address found in NEWQP, depending on the state of WRTO. The QSPI continues looping until either WREN is negated, HALT is asserted, or SPE is negated. Once WREN is negated, the QSPI finishes executing commands through the command at the address contained in ENDQP, sets the SPIF flag, and stops. When WREN is set, SPIF is set each time the QSPI transfers the entry indicated by ENDQP. 0 = Wraparound mode disabled 1 = Wraparound mode enabled
13	WRTO	Wrap to. When wraparound mode is enabled and after the end of queue has been reached, WRTO determines which address the QSPI executes next. End of queue is determined by an address match with ENDQP. Execution wraps to address 0x0 if WRTO is not set, or to the address found in NEWQP if WRTO is set.
12	—	Reserved

Table 7-17 SPCR2_B and SPCR2_A Bit Descriptions (Continued)

Bit(s)	Name	Description
11:8	ENDQP	Ending queue pointer. This field determines the last absolute address in the queue to be completed by the QSPI. After completing each command, the QSPI compares the queue pointer value of the just-completed command with the value of ENDQP. If the two values match, the QSPI assumes it has reached the end of the programmed queue and sets the SPIF flag to so indicate. The QSPI RAM queue has 16 entries: 0x0–0xF. The user may program the NEWQP to start executing commands, beginning at any of the 16 addresses. Similarly, the user may program the ENDQP to stop execution of commands at any of the 16 addresses. The queue is a circular data structure. If ENDQP is set to a lower address than NEWQP, the QSPI executes commands through address 0xF, and then continues execution at address 0x0 and so on until it stops after executing the command at address ENDQP. A maximum of 16 commands are executed before stopping, unless wraparound mode is enabled or unless the user modifies NEWQP and/or ENDQP. The user may write a NEWQP value at any time, changing the flow of execution. ENDQP may also be written at any time, changing the length of the queue. Wraparound mode may also be enabled, causing continuous execution until the mode is disabled or the QSPI is halted.
7:4	—	Reserved
3:0	NEWQP	New Queue Pointer Value. NEWQP determines which queue entry the QSPI transfers first. NEWQP should be initialized before the QSPI is enabled with SPE. NEWQP may also be written while the QSPI is operating. When this happens, the QSPI completes transfer of the queue entry in progress and then immediately begins transferring queue entries starting with the entry indicated by the NEWQP. In this way, NEWQP provides additional functionality to the QSPI by providing a mechanism for supporting multiple queues or subqueues within the QSPI RAM. By changing the value in NEWQP, the user can cause the QSPI to execute a sequence of QSPI commands beginning at any location in the queue. Therefore, the user is able to set up in advance separate subqueues for different tasks within the QSPI RAM. By writing to NEWQP, selection between the different subqueues within the QSPI RAM is accomplished. If wraparound mode is enabled by setting WREN and WRTO in SPCR2, NEWQP assumes an additional function. When the end of the queue is reached, as determined by ENDQP, the address contained in NEWQP is used by the QSPI to wrap around to the first queue entry. The QSPI then re-executes the queued commands repeatedly until halted.

7.3.4.4 QSPI Control Register 3 (SPCR3)

SPCR3 contains parameters for configuring the QSPI. The CPU can read and write this register; the QSM has read-only access.

SPCR3_B — QSPI Control Register **0xYF F41E**
SPCR3_A **0xYF FC1E**

15	14	13	12	11	10	9	8	7	0
RESERVED					LOOPQ	HMIE	HALT	SPSR*	
RESET:									
0	0	0	0	0	0	0	0		

* SPSR — QSPI Status Register

Table 7-18 SPCR3_B and SPCR3_A Bit Descriptions

Bit(s)	Name	Description
15:11	—	Reserved
10	LOOPQ	QSPI Loop mode. LOOPQ enables or disables the feedback path on the data serializer for testing. If enabled, LOOPQ routes serial output data back into the data serializer, instead of received data. If disabled, LOOPQ allows regular received data into the data serializer. LOOPQ does not affect the QSPI output pins. 0 = Feedback path disabled 1 = Feedback path enabled
9	HMIE	HALTA and MODF interrupt enable. HMIE enables or disables QSPI interrupts to the CPU caused when either the HALTA status flag or the MODF status flag in SPSR is asserted. When HMIE is set, the assertion of either flag causes the QSPI to send a hardware interrupt to the CPU. When HMIE is clear, the asserted flag does not cause an interrupt. 0 = HALTA and MODF interrupts disabled 1 = HALTA and MODF interrupts enabled
8	HALT	Halt. This bit is used by the CPU to stop the QSPI on a queue boundary. The QSPI halts in a known state from which it can later be restarted. When HALT is asserted by the CPU, the QSPI finishes executing the current serial transfer (up to 16 bits) and then halts. While halted, if the command control bit (CONT of the QSPI RAM) for the last command was asserted, the QSPI continues driving the peripheral chip-select pins with the value designated by the last command before the halt. If CONT was clear, the QSPI drives the peripheral chip-select pins to the value in QSM register PORTQS. If HALT is asserted during the last command in the queue, the QSPI completes the last command, asserts both HALTA and SPIF, and clears SPE. If the last queue command has not been executed, asserting HALT does not set SPIF nor clear SPE. QSPI execution continues when the CPU clears HALT. 0 = Halt not enabled 1 = Halt enabled
7:0	SPSR	QSPI status register. See **Table 7-19**.

7.3.4.5 QSPI Status Register (SPSR)

SPSR contains QSPI status information. Only the QSPI can assert the bits in this register. The CPU reads this register to obtain status information and writes this register to clear status flags. CPU writes to CPTQP have no effect.

SPSR_B — QSPI Status Register **0xYF F41F**
SPSR_A **0xYF FC1F**

15 … 8	7	6	5	4	3	2	1	0
SPCR3[1]	SPIF	MODF	HALTA	0	CPTQP			
RESET:								
	0	0	0	0	0	0	0	0

NOTES:
1. SPCR3 — QSPI Control Register 3.

Table 7-19 SPSR Bit Descriptions

Bit(s)	Name	Description
15:8	SPCR	QSPI control register. See Table 7-18.
7	SPIF	QSPI finished flag. SPIF is set when the QSPI finishes executing the last command determined by the address contained in ENDQP in SPCR2. When the address of the command being executed matches the ENDQP, the SPIF flag is set. The QSPI may still be outputting the last word to be transmitted when SPIF sets. If wraparound mode is enabled (WREN = 1), the SPIF is set, after completion of the command defined by ENDQP, each time the QSPI cycles through the queue. If SPIFIE in SPCR2 is set, an interrupt is generated when SPIF is asserted. Once SPIF is set, the CPU may clear it by reading SPSR followed by writing SPSR with a zero in SPIF. 0 = QSPI not finished 1 = QSPI finished
6	MODF	Mode Fault Flag. MODF is asserted by the QSPI when the QSPI is the serial master (MSTR = 1) and the slave select (PCS0/$\overline{SS}$) input pin is pulled low by an external driver. This is possible only if the PCS0/$\overline{SS}$ pin is configured as input by DDRQS. This low input to $\overline{SS}$ is not a normal operating condition. It indicates that a multimaster system conflict may exist, that another MCU is requesting to become the SPI network master, or simply that the hardware is incorrectly affecting PCS0/$\overline{SS}$. SPE in SPCR1 is cleared, disabling the QSPI. The QSPI pins revert to control by PORTQS. If MODF is set and HMIE in SPCR3 is asserted, the QSPI generates an interrupt to the CPU. The CPU may clear MODF by reading SPSR with MODF asserted, followed by writing SPSR with a zero in MODF. After correcting the mode fault problem, the QSPI can be re-enabled by asserting SPE. The PCS0/$\overline{SS}$ pin may be configured as a general-purpose output instead of input to the QSPI. This inhibits the mode fault checking function. In this case, MODF is not used by the QSPI. 0 = Normal operation 1 = Another SPI node requested to become the network SPI master while the QSPI was enabled in master mode (MSTR = 1), or the PCS0/$\overline{SS}$ pin was incorrectly pulled low by external hardware.
5	HALTA	Halt acknowledge flag. HALTA is asserted by the QSPI when it has come to an orderly halt at the request of the CPU, via the assertion of HALT. To prevent undefined operation, the user should not modify any QSPI control registers or RAM while the QSPI is halted. If HMIE in SPCR3 is set, the QSPI sends interrupt requests to the CPU when HALTA is asserted. The CPU can only clear HALTA by reading SPSR with HALTA set and then writing SPSR with a zero in HALTA. 0 = QSPI not halted 1 = QSPI halted
4	—	Reserved
3:0	CPTQP	Completed queue pointer. CPTQP contains the queue pointer value of the last command in the queue that was completed. The value of CPTQP is not updated until the command has been completed entirely. While the first command in a queue is executing, CPTQP contains either the reset value (0x0) or the pointer to the last command completed in the previous queue. If the QSPI is halted, CPTQP may be used to determine which commands have not been executed. The CPTQP may also be used to determine which locations in the receive data segment of the QSPI RAM contain valid received data.

7.3.4.6 QSPI RAM

The QSPI uses an 80-byte block of dual-access static RAM which can be accessed by both the QSPI and the CPU. Because of sharing, the length of time taken by the CPU to access the QSPI RAM when the QSPI is enabled, may be longer than when the QSPI is disabled. From one to four CPU wait states may be inserted by the QSPI in the process of reading or writing.

The size and type of access of the QSPI RAM by the CPU affects the QSPI access time. The QSPI is byte, word, and long-word addressable. Only word accesses of the RAM by the CPU are coherent accesses because these accesses are an indivisible operation. If the CPU makes a coherent access of the QSPI RAM, the QSPI cannot access the QSPI RAM until the CPU is finished. However, a long-word or misaligned word access is not coherent because the CPU must break its access of the QSPI RAM into two parts, which allows the QSPI to access the QSPI RAM between the two accesses by the CPU.

The RAM is divided into three segments: receive data RAM, transmit data RAM, and command control RAM. Receive data is information received from a serial device external to the MCU. Transmit data is information stored by the CPU for transmission to an external peripheral chip. Command control contains all the information needed by the QSPI to perform the transfer. **Figure 7-5** illustrates the organization of the RAM.

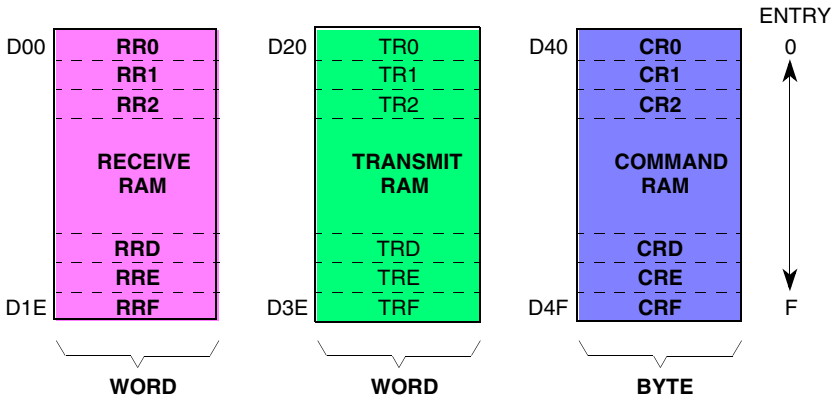


Figure 7-5 Organization of the QSPI RAM

Once the CPU has set up the queue of QSPI commands and enabled the QSPI, the QSPI operates independently of the CPU. The QSPI executes all of the commands in its queue, sets a flag indicating that it is finished, and then either interrupts the CPU or waits for CPU intervention.

The QSPI RAM's three segments are:

- Receive Data RAM
  - — This segment of the RAM stores the data that is received by the QSPI from peripherals, SPI bus masters, or other MCUs. The CPU reads this segment of RAM to retrieve the data from the QSPI. Data stored in receive RAM is right-justified, i.e., the least significant bit is always in the right-most bit position within the word (bit 0) regardless of the serial transfer length. Unused bits in a receive queue entry are set to zero by the QSPI upon completion of the individual queue entry. The CPU can access the data using byte, word, or long-word addressing.
  - — The CPTQP value in SPSR shows which queue entries have been executed.

The CPU uses this information to determine which locations in receive RAM contain valid data before reading them.

- Transmit Data RAM
  - — This segment of the RAM stores the data that is to be transmitted by the QSPI to peripherals. The CPU normally writes one word of data into this segment for each queue command to be executed. If the corresponding peripheral, such as a serial input port, is used solely to input data, then this segment does not need to be initialized.
  - — Information to be transmitted by the QSPI should be written by the CPU to the transmit data segment in a right-justified manner. The information in the transmit data segment of the RAM cannot be modified by the QSPI. The QSPI merely copies the information to its data serializer for transmission to a peripheral. Information in transmit RAM remains there until it is re-written by the CPU.
- Command RAM
  - — The command segment of the QSPI RAM is used only by the QSPI when it is in master mode. The CPU writes one byte of control information to this segment for each QSPI command to be executed. The information in the command RAM cannot be modified by the QSPI. It merely uses the information to perform the serial transfer.
  - — Command RAM consists of 16 bytes. Each byte is divided into two fields. The first, the peripheral chip-select field, activates the correct serial peripheral during the transfer. The second, the command control field, provides transfer options specifically for that command/serial transfer. This feature gives the user more control over each transfer, providing the flexibility to interface to external SPI chips with different requirements.
  - — A maximum of 16 commands can be in the queue command control bytes. These bytes are assigned an address from 0x0 – 0xF. Queue execution by the QSPI proceeds from the address contained in NEWQP through the address contained in ENDQP. Both of these fields are contained in SPCR2.

QSPI RAM(B) **0xYF F540**
QSPI RAM(A) **0xYF FD40**

7	6	5	4	3	2	1	0
CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0*
—	—	—	—	—	—	—	—
CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0[1]

QSPI RAM(B) **0xYF F54F**
QSPI RAM(A) **0xYF FD4F**

COMMAND CONTROL **PERIPHERAL CHIP-SELECT**

NOTES:
1. The PCS0 bit represents the dual-function PCS0/$\overline{SS}$.

Table 7-20 QSPI RAM Bit Descriptions

Bit(s)	Name	Description
7	CONT	Continue. Some peripheral chips must be deselected between every QSPI transfer. Other chips must remain selected between several sequential serial transfers. CONT is designed to provide the flexibility needed to handle both cases. If CONT = 1 and the peripheral chip-select pattern for the next command is the same as that of the present command, the QSPI drives the PCS pins to the same value continuously during the two serial transfers. An unlimited number of serial transfers may be sent to the same peripheral(s) without deselecting it (them) by setting CONT = 1. If CONT = 1 and the peripheral chip-select pattern for the next command is different from that of the present command, the QSPI drives the PCS pins to the new value for the second serial transfer. Although this case is similar to CONT = 0, a difference remains. When CONT = 1, the QSPI continues to drive the PCS pins using the pattern from the first transfer until it switches to using the pattern for the second transfer. When CONT = 0, the QSPI drives the PCS pins to the values found in register PORTQS between serial transfers. 0 = Return control of peripheral chip-selects to PORTQS after transfer is complete 1 = Keep peripheral chip-selects asserted after transfer is complete
6	BITSE	Bits per transfer enable. 0 = Eight bits 1 = Number of bits set in BITS field of SPCR0
5	DT	Delay after transfer. A/D converters require a known amount of time to perform a conversion. The conversion time for serial CMOS A/D converters may range from 1 – 100 µs. To facilitate interfacing to peripherals with a latency requirement, the QSPI provides a programmable delay at the end of the serial transfer, with the DT field. The user may avoid using this delay option by executing transfers with other peripheral devices in between transfers with the peripheral that requires a delay. This interleaved operation improves the effective serial transfer rate. The amount of the delay between transfers is programmable by the user via the DTL field in SPCR1. The range may be set from 1 – 489 µs at 16.78 MHz.
4	DSCK	PCS to SCK delay. 0 = PCS valid to SCK transition is 1/2 SCK 1 = DSCKL field in SPCR1 specifies value of delay from PCS valid to SCK
3:0	PCS[3:0]\	Peripheral chip-select. The four peripheral chip-select bits can be used directly to select one of four external chips for the serial transfer, or decoded by external hardware to select one of 16 chip-select patterns for the serial transfer. More than one peripheral chip-select may be activated at a time, which is useful for broadcast messages in a multinode SPI system. More than one peripheral chip may be connected to each PCS pin. Care must be taken by the system designer not to exceed the maximum drive capability of the pins. See the appropriate microcontroller user's manual for electrical specifications. QSM register PORTQS determines the state of the PCS pins when the QSPI is disabled, and also determines the state of PCS pins that are not assigned to the QSPI when the QSPI is enabled. PORTQS determines the state of pins assigned to the QSPI between transfers as well. To use a peripheral chip-select pin, the CPU assigns the pin to the QSPI in PQSPAR by writing a one to the appropriate bit. The default value of the PCS pin should be written to PORTQS. Next, the pin must be defined as an output in DDRQS by setting the appropriate bit, which causes the pin to start driving the default value. The QSPI RAM may then be initialized for a serial transmission, with the peripheral chip-select bits of the command control byte appropriately configured to activate the desired PCS pin(s) during the serial transfer. When the command is executed, the PCS pin(s) are driven to the values contained in the appropriate control byte. After completing the serial transfer, the QSPI returns control of the peripheral chip-select signal(s) (if CONT = 0 in the command control byte) to register PORTQS.

7.3.5 Operating Modes and Flowcharts

The QSPI utilizes an 80-byte block of dual-access static RAM accessible by both the QSPI and the CPU. Because of this dual access capability, up to two wait states may be inserted into CPU access times if the QSPI is in operation.

The RAM is divided into three segments: 16 command control bytes, 16 transmit data words of information to be transmitted, and 16 receive data words for data to be received. Once the CPU has a) set up a queue of QSPI commands, b) written the transmit data segment with information to be sent, and c) enabled the QSPI, the QSPI operates independently of the CPU. The QSPI executes all of the commands in its queue, sets a flag indicating completion, and then either interrupts the CPU or waits for CPU intervention.

The QSPI operates on a queue data structure contained in the QSPI RAM. Control of the queue is handled by three pointers: the new queue pointer (NEWQP), the completed queue pointer (CPTQP), and the end queue pointer (ENDQP). NEWQP, contained in SPCR2, points to the first command in the queue to be executed by the QSPI. CPTQP, contained in SPSR, points to the command last executed by the QSPI. ENDQP, also contained in SPCR2, points to the last command in the queue to be executed by the QSPI, unless wraparound mode is enabled (WREN = 1).

At reset, NEWQP is initialized to 0x0, causing QSPI execution to begin at queue address 0x0 when the QSPI is enabled (SPE = 1). CPTQP is set by the QSPI to the queue address (0x0-0xF) last executed, but is initialized to 0x0 at reset. ENDQP is also initialized to 0x0 at reset, but should be changed by the user to reflect the last queue entry to be transferred before enabling the QSPI. Leaving NEWQP and ENDQP set to 0x0 causes a single transfer to occur when the QSPI is enabled.

The organization of the QSPI RAM requires that one byte of command control data, one word of transmit data, and one word of receive data all correspond to one queue entry.

After executing the current command, ENDQP is checked against CPTQP for an end-of-queue condition. If a match occurs, the SPIF flag is set and the QSPI stops unless wraparound mode is enabled.

The QSPI operates in one of two modes: master or slave. Switching between the two operating modes is achieved under software control by writing to the master (MSTR) bit in SPCR0.

In master mode, the QSPI executes the queue of commands as defined by the control bits in each entry. Chip-select pins are activated; data is transmitted, received, and placed in the QSPI RAM.

In slave mode, a similar operation occurs in response to the slave select ($\overline{SS}$) pin activated by an external SPI bus master. The primary differences are a) no peripheral chip-selects are generated, and b) the number of bits transferred is controlled in a different manner. When the QSPI is selected, it executes the next queue transfer to correctly exchange data with the external device.

The following flowcharts, Figure 7-6, Figure 7-7, and Figure 7-11, outline the operation of the QSPI for both master and slave modes.

NOTE

The CPU must initialize the QSM global and pin registers and the QSPI control registers before enabling the QSPI for either master or slave operation. If using master mode, the necessary command control RAM should also be written before enabling the QSPI. Any data to be transmitted should also be written before the QSPI is enabled. When wrap mode is used, data for subsequent transmissions may be written at any time.

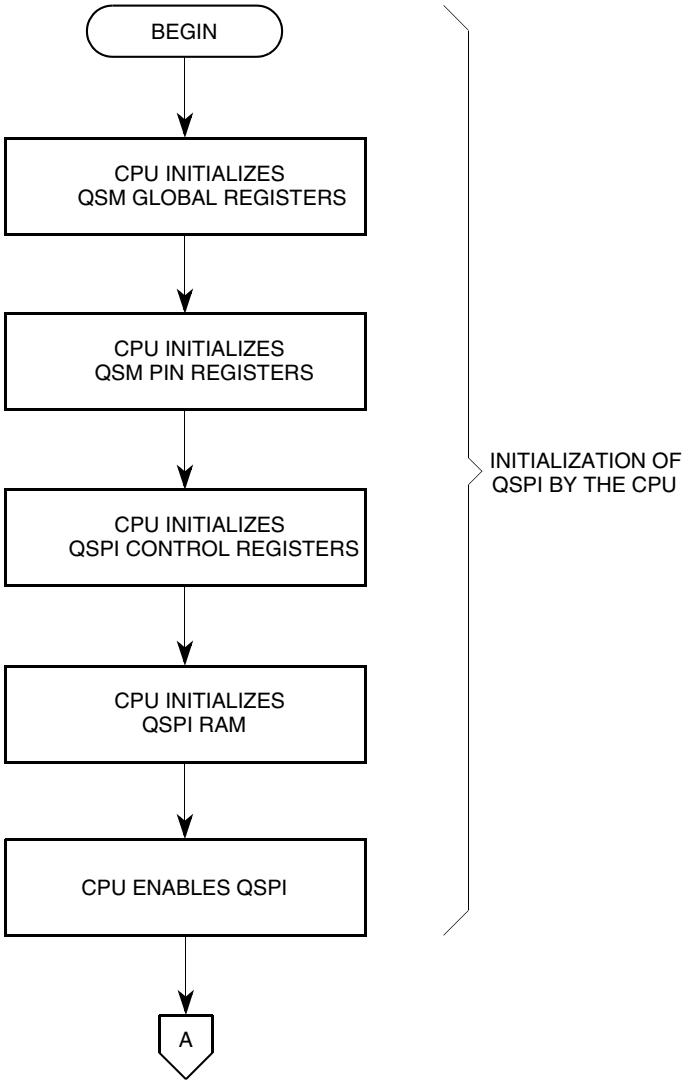


Figure 7-6 Flowchart of QSPI Initialization Operation

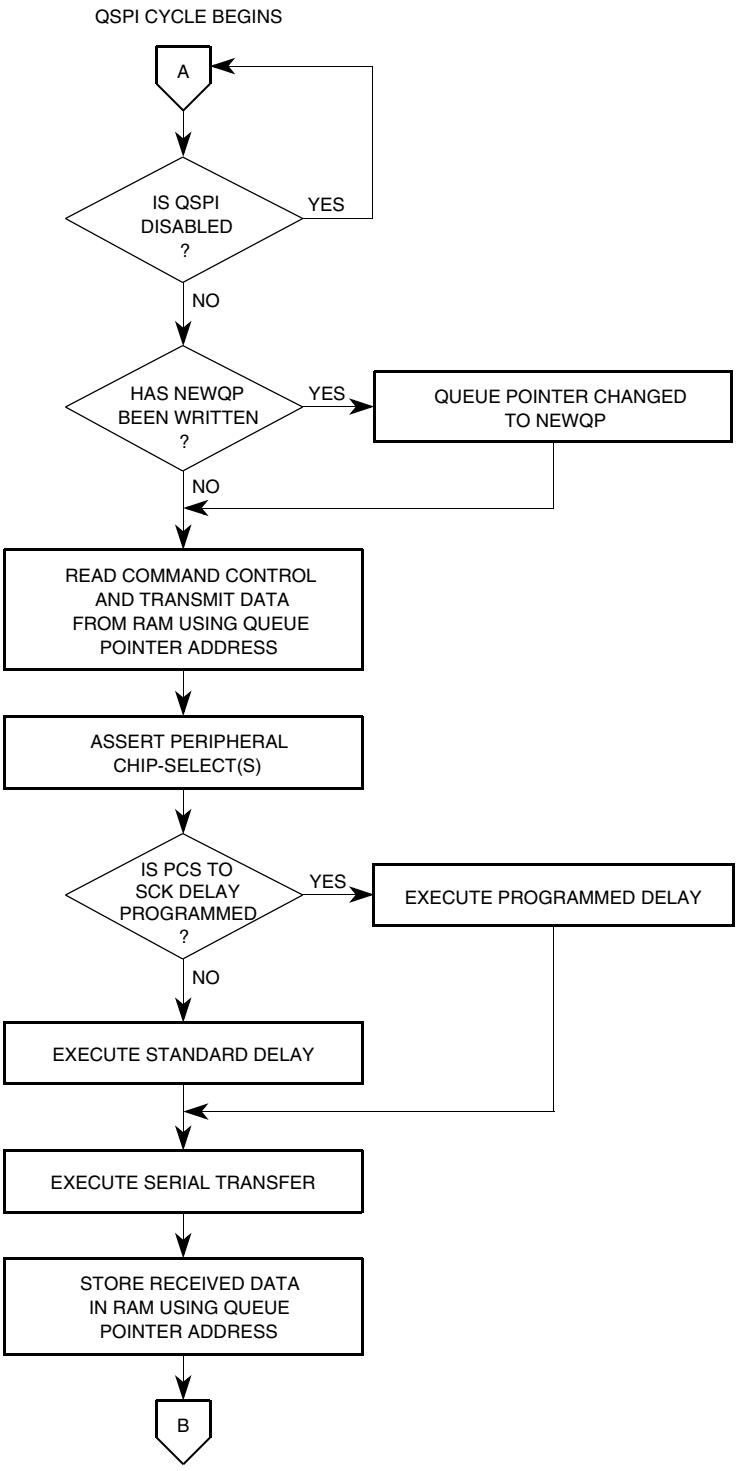


Figure 7-7 Flowchart of QSPI Master Operation (Part 1)

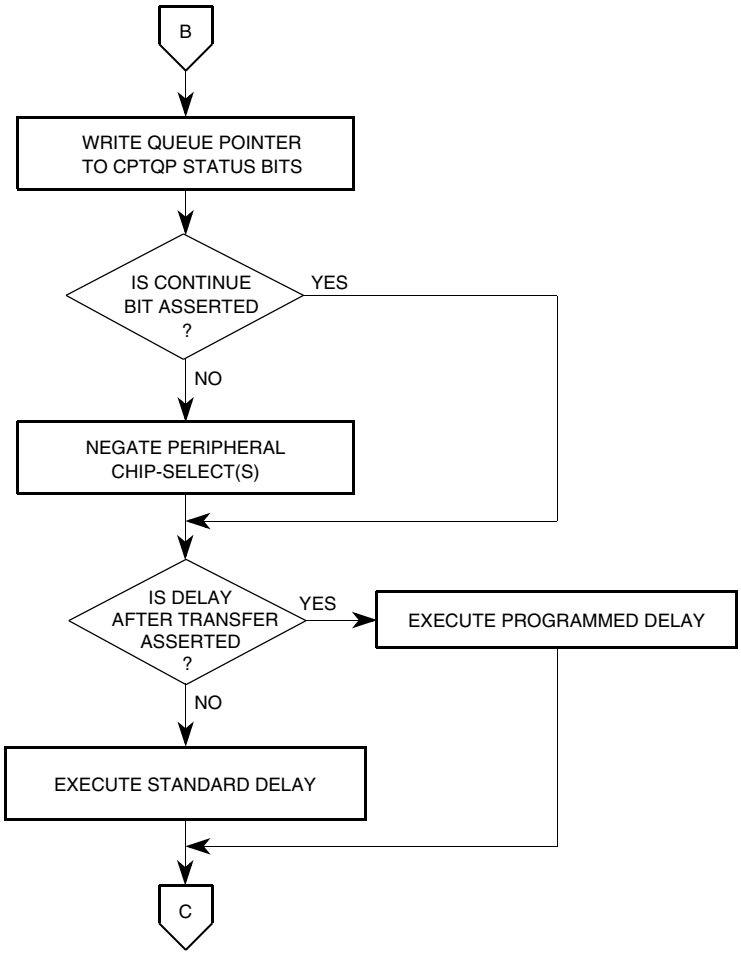


Figure 7-8 Flowchart of QSPI Master Operation (Part 2)

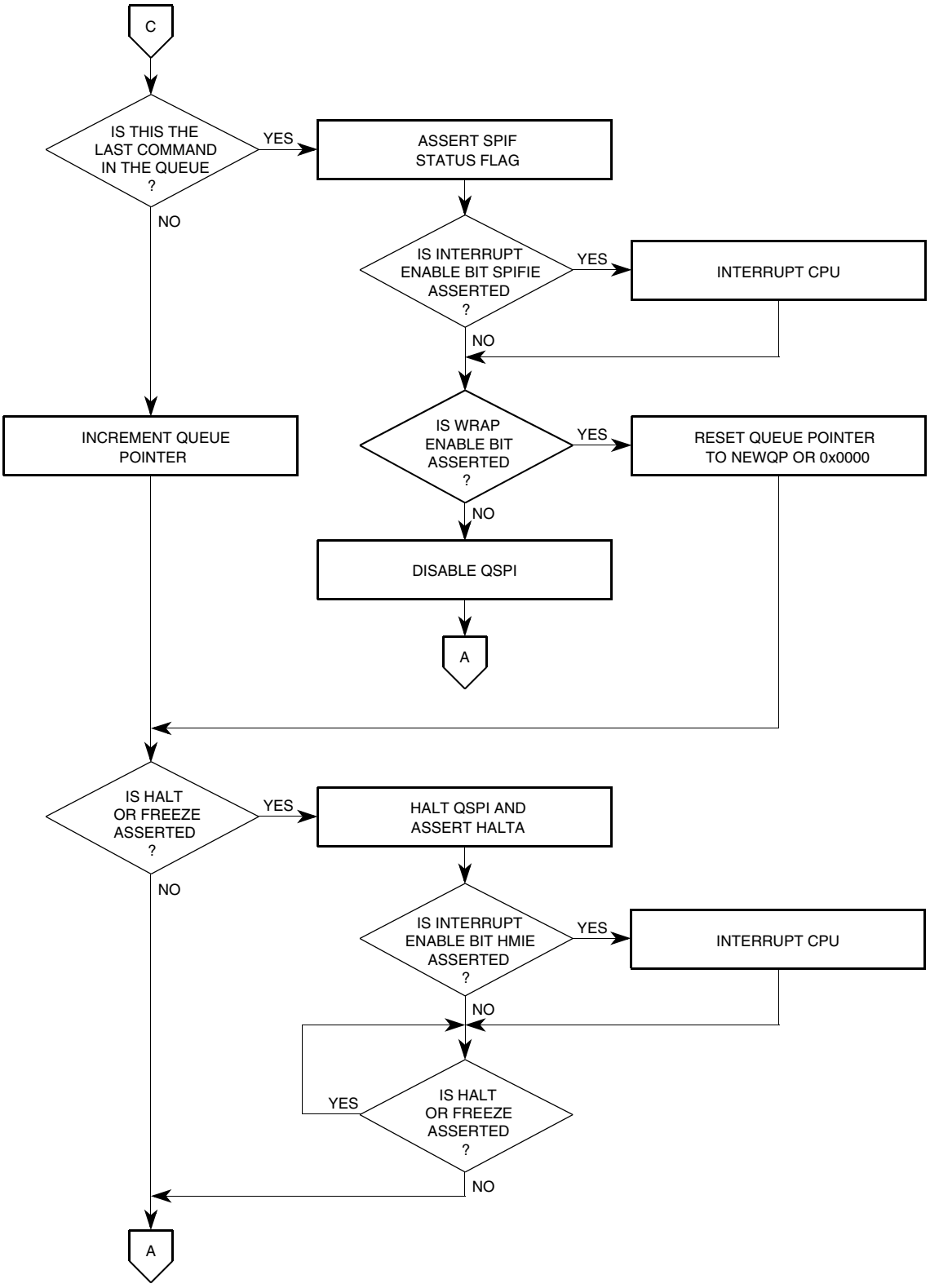


Figure 7-9 Flowchart of QSPI Master Operation (Part 3)

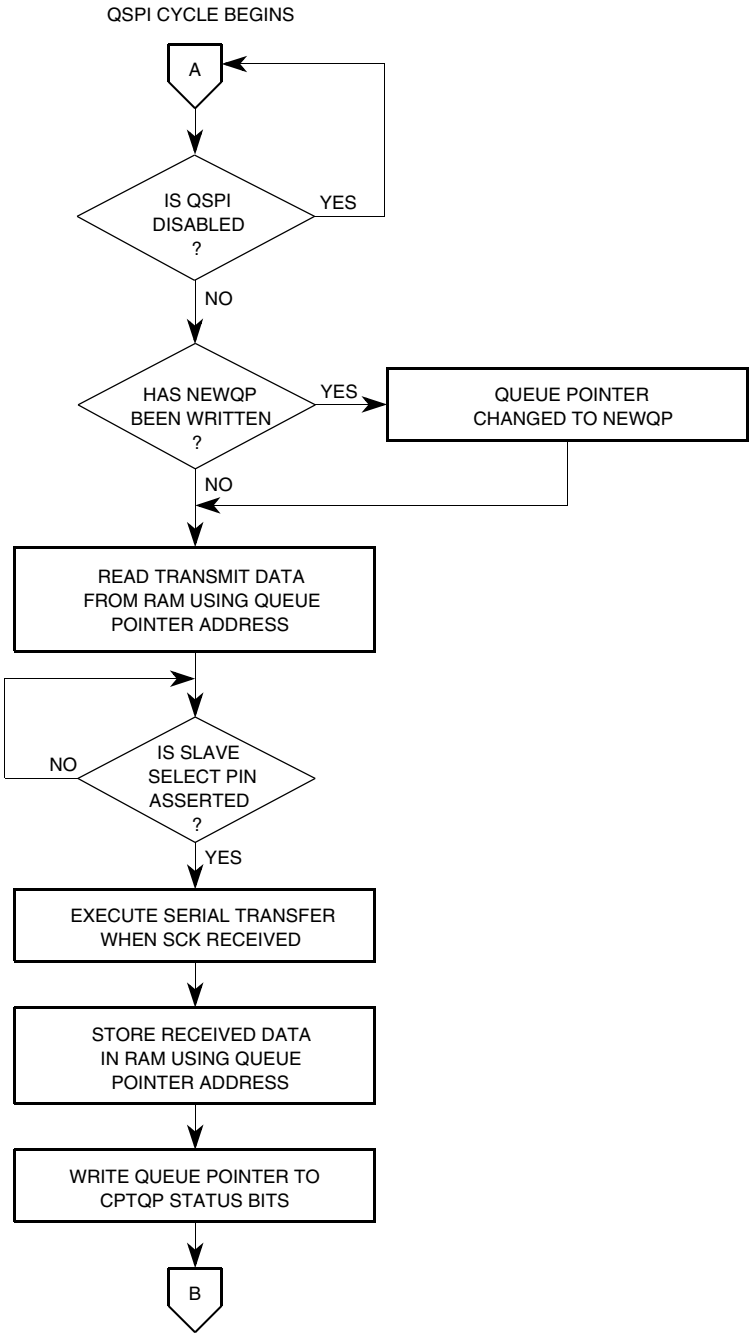


Figure 7-10 Flowchart of QSPI Slave Operation (Part 1)

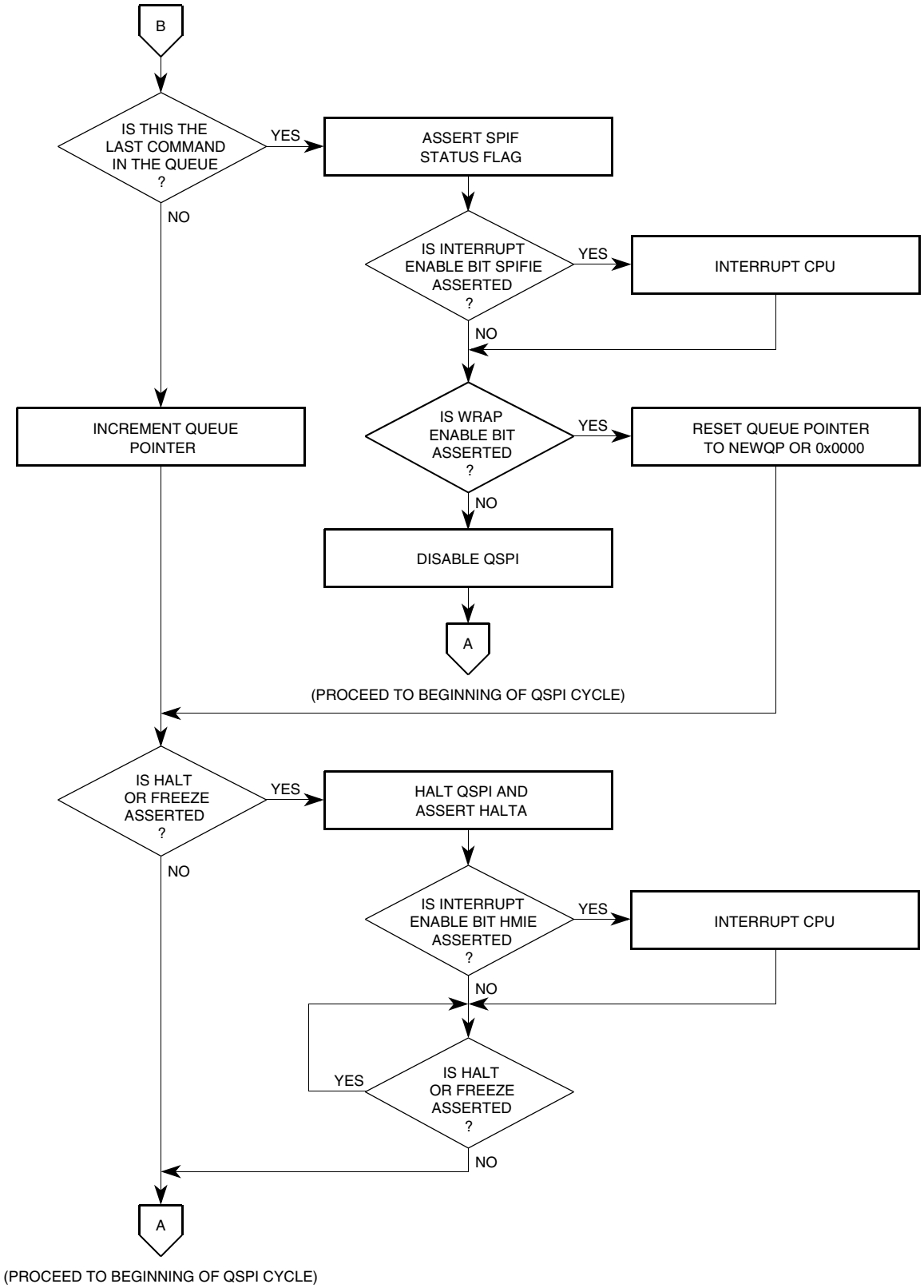


Figure 7-11 Flowchart of QSPI Slave Operation (Part 2)

Although the QSPI inherently supports multimaster operation, no special arbitration mechanism is provided. The user is given a mode fault flag (MODF) to indicate a request for SPI master arbitration; however, the system software must implement the arbitration. Note that unlike previous SPI systems, (e.g., on the M68HC11 family, MSTR is not cleared by a mode fault being set nor are the QSPI pin output drivers disabled; however, the QSPI is disabled when software clears SPE in QSPI register SPCR1).

Normally, the SPI bus performs simultaneous bidirectional synchronous transfers. The serial clock on the SPI bus master supplies the clock signal (SCK) to time the transfer of the bits. Four possible combinations of clock phase and polarity may be employed.

Data is transferred with the most significant bit first. The number of bits transferred per command defaults to eight, but may be programmed to a value from 8–16 bits, using the BITSE field.

Typically, outputs used for the SPI bus are not open-drain unless multiple SPI masters are in the system. If needed, WOMQ in SPCR0 may be set to provide open-drain outputs. An external pull-up resistor should be used on each output bus line. WOMQ affects all QSPI pins regardless of whether they are assigned to the QSPI or used as general-purpose I/O.

7.3.5.1 Master Mode

When operated in master mode, the QSPI may initiate serial transfers. The QSPI is unable to respond to any externally initiated serial transfers. QSM register DDRQS should be written to direct the data flow on the QSPI pins used. The SCK pin should be configured as an output. Pins MOSI and PCS3–PCS0/$\overline{SS}$ should be configured as outputs as necessary. MISO should be configured as an input if necessary.

QSM register PQSPAR should be written to assign the necessary bits to the QSPI. The pins necessary for master mode operation are MISO and/or MOSI, SCK, and one or more of the PCS pins, depending on the number of external peripheral chips to be selected. MISO is used as the data input pin in master mode, and MOSI is used as the data output pin in master mode. Either or both may be necessary, depending on the particular application. SCK is the serial clock output in master mode.

PCS[3:0] / $\overline{SS}$ are the select pins used to select external SPI peripheral chips for a serial transfer initiated by the QSPI. These pins operate as either active-high or active-low chip-selects. Other considerations for initialization are prescribed in **7.2.1 Overall QSM Configuration Summary**.

7.3.5.2 Master Mode Operation

After reset, the QSM registers and the QSPI control registers must be initialized as described above. In addition to the command control segment, the transmit data segment may, depending upon the application, need to be initialized. If meaningful data is to be sent out from the QSPI, the user should write the data to the transmit data segment before enabling the QSPI.

Shortly after SPE is set, the QSPI commences operation at the address indicated by NEWQP. The QSPI transmits the data found in the transmit data segment at the address indicated by NEWQP, and the QSPI stores received data in the receive data segment at the address indicated by NEWQP. Data is transferred synchronously with the internally generated SCK.

Transmit data is loaded into the data serializer (refer to **Figure 7-13**). The QSPI employs control bits, CPHA and CPOL, to determine which SCK edge the MISO pin uses to latch incoming data and which edge the MOSI pin uses to start driving the outgoing data. SPBR of SPCR0 determines the baud rate of SCK. DSCK DSCK and DSCKL determine any peripheral chip-selects valid to SCK start delay.

The number of bits transferred is determined by BITSE and BITS fields. Two options are available: the user may use the default value of eight bits or the user may program the length from 8 – 16 bits, inclusive.

Once the proper number of bits are transferred, the QSPI stores the received data in the receive data segment, stores the internal working queue pointer value in CPTQP, increments the internal working queue pointer, and loads the next data required for transfer from the queue. The internal working queue pointer address is the next command executed unless the CPU writes a new value first.

If CONT is set and the peripheral chip-select pattern does not change between the current and the pending transfer, the PCS pins are continuously driven in their designated state during and between both serial transfers. If the peripheral chip-select pattern changes, then the first pattern is driven out during execution of the first transfer, followed by the QSPI switching to the next pattern of the second transfer when execution of the second transfer begins. If CONT is clear, the deselected peripheral chip-select values (found in register PORTQS) are driven out between transfers.

DT causes a delay to occur after the specified serial transfer is completed. The length of the delay is determined by DTL. When DT is clear, the standard delay (one µs at a 16.78-MHz system clock) occurs after the specified serial transfer is completed.

7.3.5.3 Master Wraparound Mode

When the QSPI reaches the end of the queue, it always sets the SPIF flag whether wraparound mode is enabled or disabled. An optional interrupt to the CPU is generated when SPIF is asserted. At this point, the QSPI clears SPE and stops unless wraparound mode is enabled. A description of SPIFIE may be found in **7.3.4.3 QSPI Control Register 2 (SPCR2)**.

In wraparound mode, the QSPI cycles through the queue continuously. Each time the end of the queue is reached, the SPIF flag is set. If the CPU fails to clear SPIF it remains set, and the QSPI continues to send interrupt requests to the CPU (assuming SPIFIE is set). The user may avoid causing CPU interrupts by clearing SPIFIE. As SPIFIE is buffered, clearing it after the SPIF flag is asserted does not immediately stop the CPU interrupts, but only prevents future interrupts from this source. To clear the current interrupt, the CPU must read QSPI register SPSR SPSR with SPIF asserted, followed by a write to SPSR with a zero in SPIF (clear SPIF).

Execution continues in wraparound mode, even while the QSPI is requesting interrupt service from the CPU. The internal working queue pointer increments to the next address, and the commands are executed again. SPE is not cleared by the QSPI. New receive data overwrites previously received data in the receive data segment.

Wraparound mode is properly exited in two ways: a) The CPU may disable wrap-around mode by clearing WREN. The next time the end of the queue is reached, the QSPI sets SPIF, clears SPE, and stops; b) The CPU sets HALT. This second method halts the QSPI after the current transfer is completed, allowing the CPU to negate SPE. The CPU can immediately stop the QSPI by clearing SPE; however, this method is not recommended as it causes the QSPI to abort a serial transfer in process.

7.3.5.4 Slave Mode

When operating in slave mode, the QSPI may respond to externally initiated serial transfers. The QSPI is unable to initiate any serial transfers. Slave mode is typically used when multiple MCUs are in an SPI bus network, because only one device can be the SPI master (in master mode) at any given time.

QSM register DDRQS should be written to direct data flow on the QSPI pins used. The MISO and MOSI pins, if needed, should be configured as output and input, respectively. Pins SCK and PCS0/$\overline{SS}$ should be configured as inputs.

QSM register PQSPAR should be written to assign the necessary bits to the QSPI. The pins necessary for slave mode operation are MISO and/or MOSI, SCK, and PCS0/$\overline{SS}$. MISO is the data output pin in slave mode, and MOSI is the data input pin in slave mode. Either or both may be necessary depending on the particular application.

The serial clock (SCK) is the slave clock input in slave mode. PCS0/$\overline{SS}$ is the slave select pin used to select the QSPI for a serial transfer by the external SPI bus master when the QSPI is in slave mode. The external bus master selects the QSPI by driving PCS0/$\overline{SS}$ low.

When the MISO pin is configured for QSPI use (MISO bit in PQSPAR = 1) and the QSPI is set up for slave mode (MSTR bit in SPCR0 = 0) the MISO pin can be in a high-impedance state (three-stated). This occurs while the $\overline{SS}$ pin is at a logic level one. This overrides the MISO bit in the DDRQS if it is set to be an output. The MISO pin becomes active when $\overline{SS}$ is pulled low.

The command control segment is not implemented in slave mode; therefore, the CPU does not need to initialize it. This segment of the QSPI RAM and any other unused segments may be employed by the CPU as general-purpose RAM. Other considerations for initialization are prescribed in **7.2.1 Overall QSM Configuration Summary.**

7.3.5.5 Description of Slave Operation

After reset, the QSM registers and the QSPI control registers must be initialized as described above. Although the command control segment is not used, the transmit and receive data segments may, depending upon the application, need to be initial-

ized. If meaningful data is to be sent out from the QSPI, the user should write the data to the transmit data segment before enabling the QSPI.

If SPE is set and MSTR is not set, a low state on the slave select (PCS0 / $\overline{SS}$) pin commences slave mode operation at the address indicated by NEWQP. The QSPI transmits the data found in the transmit data segment at the address indicated by NEWQP, and the QSPI stores received data in the receive data segment at the address indicated by NEWQP. Data is transferred in response to an external slave clock input at the SCK pin.

Because the command control segment is not used, the command control bits and peripheral chip-select codes have no effect in slave mode operation. The QSPI does not drive any of the four peripheral chip-selects as outputs. PCS0 / $\overline{SS}$ is used as an input.

Although CONT cannot be used in slave mode, a provision is made to enable receipt of more than 16 data bits. While keeping the QSPI selected (PCS0 / $\overline{SS}$ is held low), the QSPI stores the number of bits, designated by BITS, in the current receive data segment address, increments NEWQP, and continues storing the remaining bits (up to the BITS value) in the next receive data segment address.

As long as PCS0/$\overline{SS}$ remains low, the QSPI continues to store the incoming bit stream in sequential receive data segment addresses, until either the value in BITS is reached or the end-of-queue address is used with wraparound mode disabled. When the end of the queue is reached, the SPIF flag is asserted, optionally causing an interrupt. If wraparound mode is disabled, any additional incoming bits are ignored. If wraparound mode is enabled, storing continues at either address 0x0 or the address of NEWQP, depending on the WRTO value.

When using this capability to receive a long incoming data stream, the proper delay between transfers must be used. The QSPI requires time, approximately one µs at 16.78-MHz system clock, to prefetch the next transmit RAM entry for the next transfer. Therefore, the user may select a baud rate that provides at least a one-µs delay between successive transfers to ensure no loss of incoming data. If the system clock is operating at a slower rate, the delay between transfers must be increased proportionately.

Because the BITSE option in the command control segment is no longer available, BITS sets the number of bits to be transferred for all transfers in the queue until the CPU changes the BITS value. As mentioned above, until PCS0 / $\overline{SS}$ is negated (brought high), the QSPI continues to shift one bit for each pulse of SCK. If PCS0 / $\overline{SS}$ is negated before the proper number of bits (according to BITS) is received, the QSPI, the next time it is selected, resumes storing bits in the same receive data segment address where it left off. If more than 16 bits are transferred before negating the PCS0 /$\overline{SS}$, the QSPI stores the number of bits indicated by BITS in the current receive data segment address, then increments the address and continues storing as described above. Note that PCS0/$\overline{SS}$ does not necessarily have to be negated between transfers.

Once the proper number of bits (designated by BITS) are transferred, the QSPI stores the received data in the receive data segment, stores the internal working queue pointer value in CPTQP, increments the internal working queue pointer, and loads the new transmit data from the transmit data segment into the data serializer. The internal working queue pointer address is used the next time PCS0 / $\overline{SS}$ is asserted, unless the CPU writes to the NEWQP first.

The DT and DSCK command control bits are not used in slave mode. As a slave, the QSPI does not drive the clock line nor the chip-select lines and, therefore, does not generate a delay.

In slave mode, the QSPI shifts out the data in the transmit data segment. The transmit data is loaded into the data serializer. When the PCS0 / $\overline{SS}$ pin is pulled low the MISO pin becomes active and the serializer then shifts the 16 bits of data out in sequence, most significant bit first, as clocked by the incoming SCK signal. The QSPI uses CPHA and CPOL to determine which incoming SCK edge the MOSI pin uses to latch incoming data, and which edge the MISO pin uses to drive the data out.

The QSPI transmits and receives data until reaching the end of the queue (defined as a match with the address in ENDQP), regardless of whether PCS0 / $\overline{SS}$ remains selected or is toggled between serial transfers. Receiving the proper number of bits causes the received data to be stored. The QSPI always transmits as many bits as it receives at each queue address, until the BITS value is reached or PCS0 / $\overline{SS}$ is negated.

7.3.5.6 Slave Wraparound Mode

When the QSPI reaches the end of the queue, it always sets the SPIF flag, whether wraparound mode is enabled or disabled. An optional interrupt to the CPU is generated when SPIF is asserted. At this point, the QSPI clears SPE and stops unless wraparound mode is enabled. A description of SPIFIE bit can be found in **7.3.4.3 QSPI Control Register 2 (SPCR2)**.

In wraparound mode, the QSPI cycles through the queue continuously. Each time the end of the queue is reached, the SPIF flag is set. If the CPU fails to clear SPIF, it remains set, and the QSPI continues to send interrupt requests to the CPU (assuming SPIFIE is set). The user may avoid causing CPU interrupts by clearing SPIFIE. As SPIFIE is buffered, clearing it after the SPIF flag is asserted does not immediately stop the CPU interrupts, but only prevents future interrupts from this source. To clear the current interrupt, the CPU must read QSPI register SPSR with SPIF asserted, followed by a write to SPSR with zero in SPIF (clear SPIF). Execution continues in wraparound mode even while the QSPI is requesting interrupt service from the CPU. The internal working queue pointer is incremented to the next address and the commands are executed again. SPE is not cleared by the QSPI. New receive data overwrites previously received data located in the receive data segment.

Wraparound mode is properly exited in two ways: a) The CPU may disable wraparound mode by clearing WREN. The next time end of the queue is reached, the QSPI sets SPIF, clears SPE, and stops; and, b) The CPU sets HALT. This second method

halts the QSPI after the current transfer is completed, allowing the CPU to negate SPE. The CPU can immediately stop the QSPI by clearing SPE; however, this method is not recommended, as it causes the QSPI to abort a serial transfer in process (i.e., the SPI stops the transmission currently in progress).

7.4 SCI Submodule

The SCI submodule is used to communicate with external devices and other MCUs via an asynchronous serial bus. The SCI is fully compatible with the SCI systems found on other Motorola MCUs, such as the M68HC11 and M68HC05 families. It has all of the capabilities of previous SCI systems as well as several significant new features. The following paragraphs describe the features, pins, programmer's model (memory map), registers, and the transmit and receive operations of the SCI.

7.4.1 Features

Standard SCI features are listed below, followed by a list of additional features offered.

Standard SCI two-wire system features:

- Standard nonreturn-to-zero (NRZ) mark / space format
- Advanced error detection mechanism (detects noise duration up to 1/16 of a bit-time)
- Full-duplex operation
- Software selectable word length (8- or 9-bit words)
- Separate transmitter and receiver enable bits
- May be interrupt driven
- Four separate interrupt enable bits

Standard SCI receiver features:

- Receiver wakeup function (idle or address mark bit)
- Idle-line detect
- Framing error detect
- Noise detect
- Overrun detect
- Receive data register full flag

Standard SCI transmitter features:

- Transmit data register empty flag
- Transmit complete flag
- Send break

QSM-enhanced SCI two-wire system features:

- 13-bit programmable baud-rate modulus counter
  - — A baud rate modulus counter has been added to provide the user with more flexibility in choosing the crystal frequency for the system clock. The modulus counter allows the SCI baud rate generator to produce standard transmission frequencies for a wide range of system clocks. The user is no longer con-

strained to select crystal frequencies based on the desired serial baud rate. This counter provides baud rates from 64 baud to 524 kbaud with a 16.78-MHz system clock.

- Even/odd parity generation and detection
  - — The user now has the choice either of seven or eight data bits plus one parity bit, or of eight or nine data bits with no parity bit. Even or odd parity is available. The transmitter automatically generates the parity bit for a transmitted byte. The receiver detects when a parity error has occurred on a received byte and sets a parity error flag.

QSM-enhanced SCI receiver features:

- Two idle-line detect modes
  - — Standard Motorola SCI systems detect an idle line when 10 or 11 consecutive bit-times are all ones. Used with the receiver wakeup mode, the receiver can be awakened prematurely if the message preceding the start of the idle line contained ones in advance of its stop bit. The new (second) idle-line detect mode starts counting idle time only after a valid stop bit is received, which ensures correct idle-line detection.
- Receiver active flag (RAF)
  - — RAF indicates the status of the receiver. It is set when a possible start bit is detected and is cleared when an idle line is detected. RAF is also cleared if the start bit is determined to be line noise. This flag can be used to prevent collisions in systems with multiple masters.

7.4.2 SCI Programmer's Model and Registers

The programmer's model (memory map) for the SCI submodule consists of the QSM global and pin control registers (refer to **7.2.2 QSM Global Registers** and **7.2.3 QSM Pin Control Registers**) and the four SCI registers. The SCI registers are listed in **Table 7-21** and consist of two control registers, one status register, and one data register. All registers may be read or written at any time by the CPU. Rewriting the same value to any SCI register does not disrupt operation; however, writing a different value into an SCI register when the SCI is running may disrupt operation. To change register values, the receiver and transmitter should be disabled with the transmitter allowed to finish first. The status flags in register SCSR may be cleared at any time.

Table 7-21 SCI Register

Address	Name	Usage
0xYF F408_B 0xYF FC08_A	SCCR0	SCI Control Register 0
0xYF F40A_B 0xYF FC0A_A	SCCR1	SCI Control Register 1
0xYF F40C_B 0xYF FC0C_A	SCSR	SCI Status Register
0xYF F40E_B 0xYF FC0E_A	SCDR	SCI Data Register Transmit Data Register (TDR)* Receive Data Register (RDR)*

*Reads access the RDR; writes access the TDR.

When initializing the SC, the SCCR1 has two bits that should be written last; the transmitter enable (TE) and receiver enable (RE) bits, which enable the SCI. Registers SCCR0 and SCCR1 should both be initialized at the same time or before TE and RE are asserted. A single word write to SCCR1 can be used to initialize the SCI and enable the transmitter and receiver.

Figure 7-12 and **Figure 7-13** show the block diagrams for the SCI receiver and transmitter.

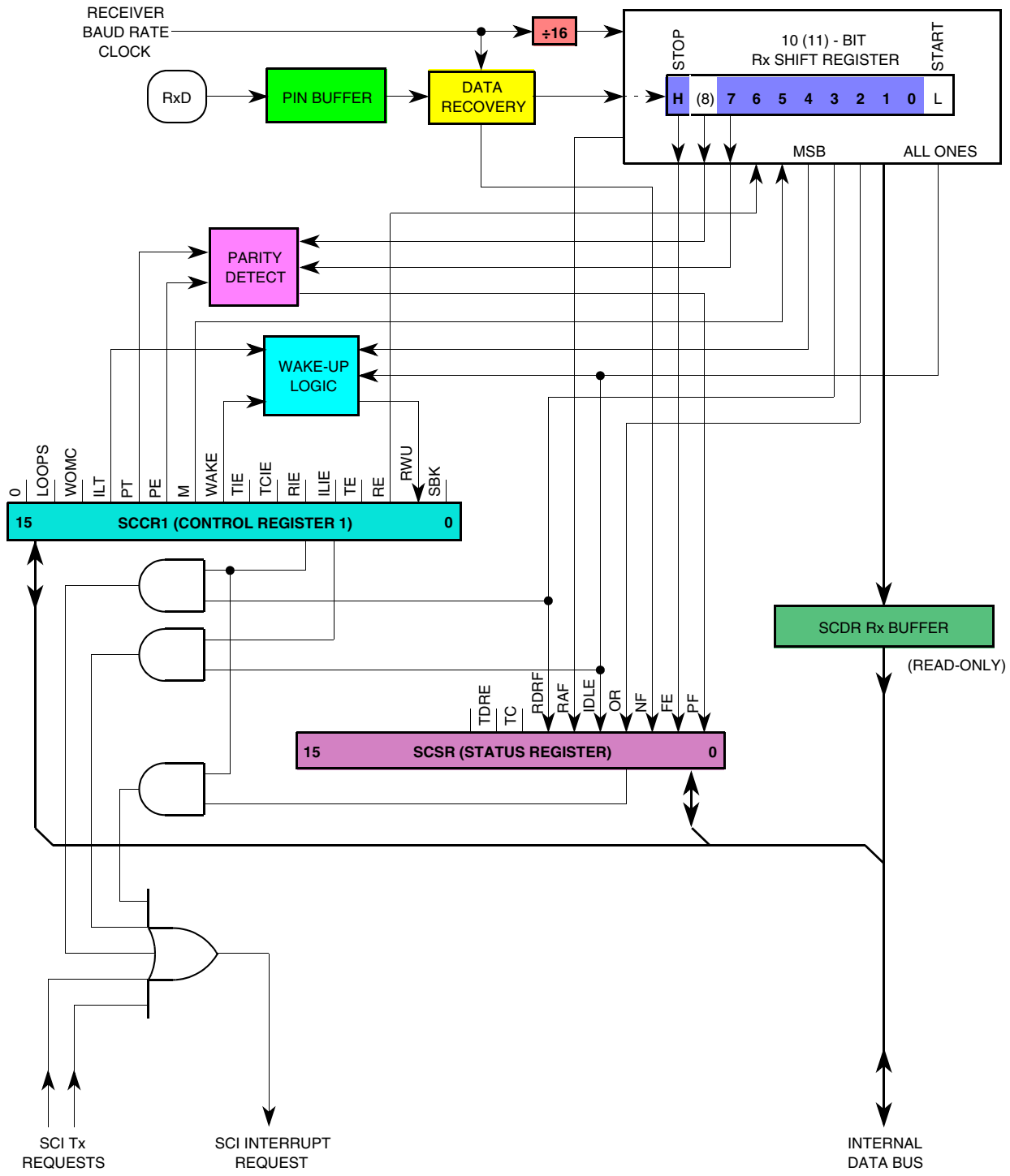


Figure 7-12 SCI Receiver Block Diagram

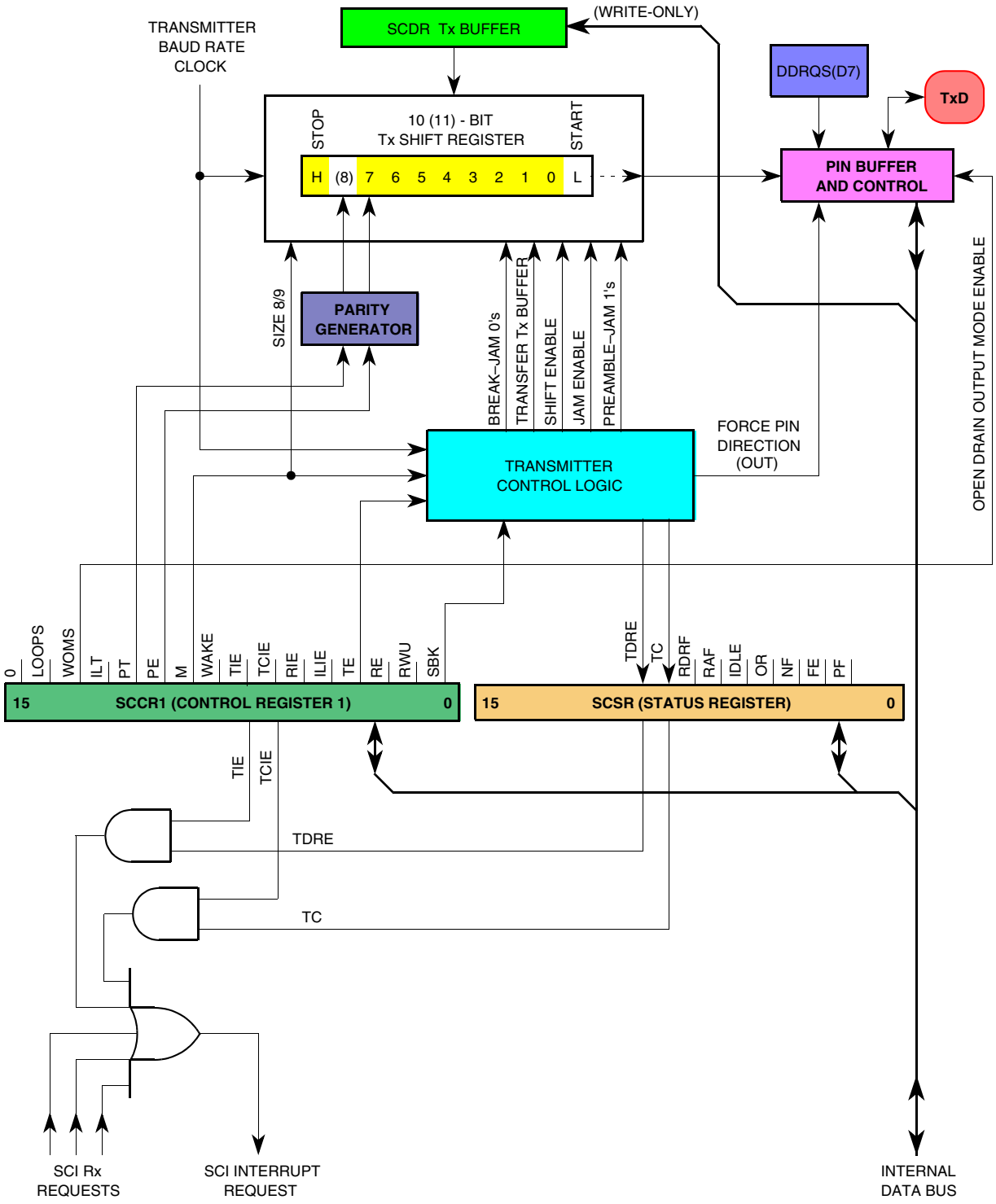


Figure 7-13 SCI Transmitter Block Diagram

7.4.2.1 SCI Control Register 0 (SCCR0)

SCCR0 contains the parameter for configuring the SCI baud rate. The baud rate should be set before the SCI is enabled. The CPU can read and write this register at any time.

SCCR0 — SCI Control Register 0 **0xYF F408**
0xYF FC08

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			SCBR												
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Table 7-22 SCCR0 Bit Descriptions

Bit(s)	Name	Description
15:13	—	Reserved
12:0	SCBR	Baud rate. The SCI baud rate is programmed by writing a 13-bit value to SCBR and is derived from the MCU system clock using a modulus counter. The SCI receiver operates asynchronously. Therefore, the SCI requires an internal clock to synchronize itself to the incoming data stream. The SCI baud-rate generator produces a receiver sampling clock with a frequency 16 times that of the expected baud rate of the incoming data. From transitions within the received waveform, the SCI determines the most likely position of the bit boundaries and adjusts sampling points to the proper positions within the bit period. The receiver sampling rate is always 16 times the frequency of the SCI baud rate, which is calculated using the following equation: SCI Baud = System Clock / (32 * SCBR) where SCBR equals {1, 2, 3,... 8191}. Note that zero is a disallowed value for SCBR. Writing a value of zero to SCBR disables the baud rate generator. There are 8191 different bauds available. The baud value depends on the value for SCBR and the system clock, as used in the above equation. Table 7-23 shows possible baud rates for a 16.78-MHz system clock. The maximum baud rate with this system clock speed is 524 Kbaud.

Table 7-23 Examples of SCI Baud Rates

Rates based on a 16.78-MHz system clock.

Nominal Baud Rate	Actual Baud Rate	Percent Error	Value of SCBR
500,000.00	524,288.00	4.86	1
38,400.00	37,449.14	–2.48	14
32,768.00	32,768.00	0.00	16
19,200.00	19,418.07	1.14	27
9,600.00	9,532.51	–0.70	55
4,800.00	4,809.98	0.21	109
2,400.00	2,404.99	0.21	218
1,200.00	1,199.74	–0.02	437
600.00	599.87	–0.02	874
300.00	299.94	–0.02	1,748
110.00	110.01	0.01	4,766
64.00	64.00	0.01	8,191

More accurate baud rates can be obtained by varying the system clock frequency with the VCO synthesizer. Each VCO speed increment adjusts the baud rate up or down by 1/64; or 1.56%.

7.4.2.2 SCI Control Register 1 (SCCR1)

SCCR1 contains parameters for configuration of the SCI. The CPU can read and write this register at any time. The SCI may modify the RWU bit in some circumstances. In general, the interrupts enabled by these control bits are cleared by reading the status register SCSR, followed by reading (for receiver status bits) or by writing (for transmitter status bits) the data register SCDR. For further detail refer to **7.4.3 Transmitter Operation** and **7.4.4 Receiver Operation**, respectively.

SCCR1 — SCI Control Register 1 **0xYF F40A**
0xYF FC0A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RE-SERVED	LOOPS	WOMS	ILT	PT	PE	M	WAKE	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 7-24 SCCR1 Bit Descriptions

Bit(s)	Name	Description
15	—	Reserved
14	LOOPS	LOOP mode. LOOPS controls a feedback path on the data serial shifter. If enabled, the output of the SCI transmitter is fed back into the receive serial shifter as receiver input, and no data is driven out of the TXD pin nor is data received from the RXD pin. The TXD pin is driven high (idle line). Both the transmitter and receiver must be enabled for loop mode to function. 0 = Normal SCI operation, no looping, feedback path disabled 1 = Test SCI operation, looping, feedback path enabled
13	WOMS	Wired-OR mode for SCI pins. WOMS determines whether the TXD pin is an open-drain output or a normal CMOS output. This bit is used only when TXD is an output. If the TXD pin is being used as a general-purpose input pin, WOMS has no effect. 0 = If configured as an output, TXD is a normal CMOS output. 1 = If configured as an output, TXD is an open-drain output.
12	ILT	Idle-line detect type. ILT determines which one of two types of idle-line detection is to be used by the SCI receiver. The short idle-line detection circuitry causes the SCI receiver to start counting ones at any point (even during the frame), which means that the stop bit and any contiguous one data bits at the end of the last byte are counted toward the 10 or 11 ones in an idle frame. Hence, the data content of the last byte transmitted may affect the timing of idle-line detection. The long idle-line detection circuitry causes the SCI receiver to start counting ones right after a stop bit, which means that the stop bit and any contiguous one data bits in a previous data byte are not counted toward the 10 or 11 ones in an idle line. Hence, the data content of the last byte transmitted does not affect the timing of idle-line detection. 0 = Short idle-line detect (starts counting when the first one is received) 1 = Long idle-line detect (starts counting when the first one is received after a stop bit(s))
11	PT	Parity type. If the data contains an even number of ones, then the parity bit equals zero. If the data contains an odd number of ones, then the parity bit equals one. When parity is enabled, PT determines whether parity is even or odd for both the receiver and the transmitter. If the data contains an even number of ones, then the parity bit equals one. If the data contains an odd number of ones, then the parity bit equals zero. 0 = Even parity 1 = Odd parity

Table 7-24 SCCR1 Bit Descriptions (Continued)

Bit(s)	Name	Description
10	PE	Parity enable. PE determines whether parity is enabled or disabled for both the receiver and the transmitter. If PE is set, the transmitter internally generates the parity bit and appends it to the data bits during transmission. The receiver checks the last bit before a stop bit to determine if the correct parity was received. If the received parity bit is not correct, the SCI sets the PF error flag in SCSR. When PE is set, the most significant bit (MSB) of the data field is used for the parity function, which results in either seven or eight bits of user data, depending on the condition of M bit. **Table 7-25** lists the available choices. 0 = SCI parity disabled 1 = SCI parity enabled; the transmitter generates the parity bit and the receiver checks incoming parity.
9	M	Mode select. The M bit determines the SCI frame format. If M is clear (its reset value), the frame format is one start bit, eight data bits, one stop bit. If M is set, the frame format is one start bit, nine data bits, one stop bit. The ninth data bit can be controlled by software to perform a function such as address mark. Frames with the ninth data bit set could be identified as an address mark. All receivers in a network could be placed in wakeup mode until an address mark is detected, at which time all receivers would wake up and read the address. All receivers being addressed could continue to receive the following message, while all receivers not being addressed could be put back into wakeup mode. The ninth data bit could also serve as a second stop bit. By setting this bit permanently to one, communication with other SCIs requiring two stop bits could be accommodated. Only 10 or 11 bits in a frame are allowed. If parity is to be enabled, the last data bit must be used for this purpose. The parity bit may be odd, even, mark, or space. Parity and address (control) bits are mutually exclusive. A choice must be made between one or the other, or neither. Every frame must have one start bit and at least one stop bit. The possible combinations are given in the bit description of PE. 0 = SCI frame: one start bit, eight data bits, one stop bit (ten bits total) 1 = SCI frame: one start bit, nine data bits, one stop bit (eleven bits total)
8	WAKE	Wakeup by address mark. WAKE determines which one of two conditions wakes up the SCI receiver when it is in wakeup mode. If WAKE is clear (its reset value), the detection of an idle line (10 or 11 contiguous ones) which clears RWU causes the SCI receiver to wake up. If WAKE is set, the detection of an address mark (the last data bit of a frame is set) which clears RWU causes the SCI receiver to wake up. 0 = SCI receiver awakened by idle-line detection 1 = SCI receiver awakened by address mark (eighth or ninth (last) bit set)
7	TIE	Transmit interrupt enable. When set, TIE enables an SCI interrupt whenever the TDRE flag in SCSR is set. The interrupt is blocked by negating TIE. 0 = SCI TDRE interrupts inhibited 1 = SCI TDRE interrupts enabled
6	TCIE	Transmit complete interrupt enable. When set, TCIE enables an SCI interrupt whenever the TC flag in SCSR is set. The interrupt may be cleared by reading SCSR when TC is set and then by writing the transmit data register (TDR) of SCDR. The interrupt is blocked by negating TCIE. 0 = SCI TC interrupts inhibited 1 = SCI TC interrupts enabled
5	RIE	Receiver interrupt enable. When set, RIE enables an SCI interrupt whenever the RDRF flag in SCSR is set. The interrupt is blocked by negating RIE. 0 = SCI RDRF interrupts inhibited 1 = SCI RDRF interrupts enabled
4	ILIE	ILIE — Idle-line interrupt enable. When set, ILIE enables an SCI interrupt whenever the IDLE flag in SCSR is set. The interrupt is blocked by negating ILIE. 0 = SCI IDLE interrupts inhibited 1 = SCI IDLE interrupts enabled

Table 7-24 SCCR1 Bit Descriptions (Continued)

Bit(s)	Name	Description
3	TE	Transmitter enable. When set, TE enables the SCI transmitter and assigns it to the TXD pin. When TE is clear, the TXD pin may be used for general-purpose I/O. An idle frame, called a preamble, consisting of ten (or eleven) contiguous ones, is automatically transmitted whenever TE is changed from zero to one. Refer to **7.4.3 Transmitter Operation** for a detailed description of TE and the SCI transmit operation. 0 = SCI transmitter disabled, TXD pin may be used as general-purpose I/O 1 = SCI transmitter enabled, TXD pin dedicated to the SCI transmitter
2	RE	Receiver enable. RE enables the SCI receiver when set. When disabled, the receiver status bits RDRF, IDLE, OR, NF, FE, and PF are inhibited and are not asserted by the SCI. Refer to **7.4.4 Receiver Operation** for a complete description of RE and the SCI receiver operation. 0 = SCI receiver disabled 1 = SCI receiver enabled
1	RWU	Receiver wakeup. Setting RWU enables the wakeup function, which allows the SCI to ignore received data until awakened by either an idle line or address mark (as determined by WAKE). When in wakeup mode, the receiver status flags are not set, and interrupts are inhibited. This bit is cleared automatically (returned to normal mode) when the receiver is awakened. 0 = Normal receiver operation, all received data recognized 1 = Wakeup mode enabled, all received data ignored until awakened
0	SBK	Send break. SBK provides the ability to transmit a break code (ten or eleven contiguous zeros) from the SCI. When SBK is set, the SCI completes the current frame transmission (if it is transmitting) and then begins transmitting continuous frames of 10 (or 11) zeros until SBK is cleared. If SBK is toggled by writing it first to a one and then immediately to a zero (in less than one serial frame interval), the transmitter sends only one or two break frames before reverting to mark (idle line) or before commencing to send data. SBK is normally used to broadcast the termination of a transmission. 0 = Normal operation 1 = Break frame(s) transmitted after completion of the current frame

Table 7-25 M and PE Bit Fields

M	PE	Result
0	0	8 Data Bits
0	1	7 Data Bits, 1 Parity Bit
1	0	9 Data Bits
1	1	8 Data Bits, 1 Parity Bit

7.4.2.3 SCI Status Register (SCSR)

SCSR contains flags that the SCI sets to inform the user of various operational conditions. These flags are automatically cleared either by hardware or by a special acknowledgment sequence consisting of an SCSR read (either the upper byte, the lower byte, or the entire word) with a flag bit set, followed by a read (or write in the case of flags TDRE and TC) of data register SCDR (either the lower byte, or the entire word). An upper byte access of SCDR is only meaningful for reads.

NOTE

A long-word read can consecutively access both registers SCSR and SCDR. This action clears the receive status flag bits that were set at the time of the read, but does not clear the TDRE or TC flags. To clear TDRE or TC, the SCSR read must be followed by a write to register SCDR (either the lower byte or the entire word).

If an internal SCI signal for setting a status bit comes after the CPU has read the asserted status bits but before the CPU has written or read register SCDR, the newly set status bit is not inadvertently cleared. Instead, register SCSR must be read again with the status bit set, and register SCDR must be written or read before the status bit is cleared.

NOTE

None of the status bits are cleared by reading a status bit while it is asserted and then by writing zero to that same bit. The procedure outlined above must be followed. Emphasis is also given to note that reading either byte of register SCSR causes all 16 bits to be accessed, and any status bits already set in either byte are armed to clear on a subsequent read or write of register SCDR.

As mentioned, register SCSR co-functions with register SCDR. SCDR is a combination of two data registers: the TDR and the RDR. Each of these data registers has a serial shifter.

SCSR(B) — SCI Status Register **0xYF F40C**
SCSR(A) **0xYF FC0C**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							TDRE	TC	RDRF	RAF	IDLE	OR	NF	FE	PF
RESET:															
0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

Table 7-26 SCSR Bit Descriptions

Bit(s)	Name	Description
15:9	—	Reserved
8	TDRE	Transmit data register empty flag. TDRE is set when the byte in register TDR is transferred to the transmit serial shifter. If this bit is zero, the transfer is yet to occur and a write to TDR will overwrite the previous value. New data is not transmitted if TDR is written without first clearing TDRE, which is accomplished by reading register SCSR with TDRE set, followed by a write to TDR. Reset sets this bit. 0 = Register TDR still contains data to be sent to the transmit serial shifter 1 = A new character may now be written to register TDR
7	TC	Transmit complete flag. TC is set when the transmitter finishes shifting out all data, queued preambles (mark/idle line), or queued breaks (logic zero). TC is cleared when SCSR is read with TC set, followed by a write to register TDR. 0 = SCI transmitter is busy 1 = SCI transmitter is idle

Table 7-26 SCSR Bit Descriptions (Continued)

Bit(s)	Name	Description
6	RDRF	Receive data register full flag. RDRF is set when the content of the receive serial shifter is transferred to register RDR. If one or more errors are detected in the received word, the appropriate receive-related flag(s) NF, FE, and/or PF are set within the same clock cycle. RDRF is cleared when register SCSR is read with RDRF set, followed by a read of register RDR. 0 = Register RDR is empty or contains previously read data 1 = Register RDR contains new data
5	RAF	Receiver active flag. RAF indicates whether the SCI receiver is busy. This flag is set when the SCI receiver detects a possible start bit and is cleared when the chosen type of idle line is detected. RAF can be used to reduce collisions in systems with multiple masters. The SCI receiver samples each start bit 16 times (at a rate of 16 times the baud rate). The 16 sample times are called RT1–RT16. RAF is set initially at RT1. The SCI receiver samples RT3, RT5, and RT7. If the receiver line is high during two or three of the three receive time (RT) samples, the start bit is considered invalid, and RAF is subsequently cleared. A more detailed description is found in **7.4.4.1 Receiver Bit Processor**. 0 = SCI receiver is idle 1 = SCI receiver is busy
4	IDLE	Idle-line detected flag. DLE is set when the SCI receiver detects an idle-line condition (reception of a minimum of ten or eleven consecutive ones as specified by ILT in SCCR1). This bit is not set by the idle-line condition when RWU in SCCR1 is set. Once cleared, IDLE is not set again until after RDRF is set (after the line is active and becomes idle again). If a break is received, RDRF is set, allowing a subsequent idle line to be detected again. IDLE is cleared when SCSR is read with IDLE set, followed by a read of register RDR. Under certain conditions, the IDLE flag may be set immediately following the negation of RE (SCCR1). System designs should ensure this causes no detrimental effects. 0 = SCI receiver did not detect an idle-line condition 1 = SCI receiver detected an idle-line condition
3	OR	Overrun error flag. OR is set when a new byte is ready to be transferred from the receive serial shifter to register RDR, RDR is already full (RDRF is still set), and a new start bit is detected. Data transfer is inhibited until OR is cleared. Previous data in RDR remains valid, but additional data received during an overrun condition (including the byte that set OR) is lost. A difference exists between OR and the other receiver status flags, NF, FE, and PF all reflect the status of data already transferred to register RDR. OR reflects an operational condition that resulted in a loss of data to RDR. OR is cleared when SCSR is read with OR set, followed by a read of register RDR. 0 = No overrun has occurred 1 = An overrun condition has occurred indicating that at least one byte of data was overwritten in the receive serial shifter
2	NF	Noise Error Flag. NF is set when the SCI receiver detects noise on a valid start bit, on any of the data bits, or on the stop bit(s). It is not set by noise on the idle line or on invalid start bits. Each bit is sampled three times for noise. If the three samples are not at the same logic level, the majority value is used for the received data value, and NF is set. NF is not set until the entire frame is received and RDRF is set. Although an interrupt is not explicitly associated with NF, an interrupt may be generated with RDRF and NF checked in this manner. NF is cleared when SCSR is read with NF set, followed by a read of register RDR. 0 = No noise detected on the received data 1 = Noise occurred on the received data

Table 7-26 SCSR Bit Descriptions (Continued)

Bit(s)	Name	Description
1	FE	Framing error flag. FE is set when the SCI receiver detects a zero where a stop bit (one) was to occur. A framing error results when the frame boundaries in the received bit stream are not synchronized with the receiver bit counter. FE is not set until the entire frame is received and RDRF is set. Although an interrupt is not explicitly associated with FE, an interrupt may be generated with RDRF and FE checked in this manner. A break can also cause FE to be set. FE is cleared when SCSR is read with FE set, followed by a read of register RDR. 0 = No framing error on the received data 1 = Framing error or break occurred on the received data
0	PF	Parity error flag. PF is set when the SCI receiver detects a parity error. PF is not set until the entire frame is received and RDRF is set. Although an interrupt is not explicitly associated with PF, an interrupt may be generated with RDRF and PF checked in this manner. PF is cleared when SCSR is read with PF set, followed by a read of the register RDR. 0 = No parity error occurred on the received data 1 = Parity error occurred on the received data

7.4.2.4 SCI Data Register (SCDR)

SCDR contains two data registers, both at the same address. The first register is the RDR, which is a read-only register. It contains data received over the SCI serial interface. Initially, data is received into the receive serial shifter and is transferred by the receiver into RDR. The second register is the SCI TDR, which is a write-only register. Data to be transmitted over the SCI serial interface is written to TDR. The transmitter transfers this data to the transmit serial shifter, adding on additional format bits before the data is sent out on the SCI serial interface.

SCDR(B) — SCI Data Register **0xYF F40E**
SCDR(A) **0xYF FC0E**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							R8/T8	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0
RESET:															
0	0	0	0	0	0	0	U	U	U	U	U	U	U	U	U

Table 7-27 SCDR Bit Descriptions

Bit(s)	Name	Description
15:9	—	Reserved
8	R8/T8	Receive 8/transmit 8. This bit is the ninth serial data bit received (R8) when the SCI system is configured for a 9-bit data operation (M = 1). When the SCI system is configured for an 8-bit data operation (M = 0), this bit has no meaning or effect. This bit is the ninth serial data bit transmitted (T8) when the SCI system is configured for 9-bit data operation (M = 1). When the SCI system is configured for an 8-bit data operation (M = 0), this bit has no meaning or effect. Accesses to the lower byte of SCDR triggers the mechanism for clearing the status bits or for initiating transmissions whether byte, word, or long-word accesses are used.
7:0	R[7:0]/ T[7:0]	Receive[7:0]/transmit[7:0]. The first eight bits (7:0) contain the first eight data bits to be received (R[7:0]) when SCDR is read, and also contain the first eight data bits to be transmitted (T[7:0]) when SCDR is written.

7.4.3 Transmitter Operation

The transmitter consists of a transmit serial shifter and a parallel transmit data register (TDR) located in SCDR (refer to 7.4.2.4 SCI Data Register (SCDR)). A character may be loaded into the TDR while another character is being shifted out, a capability called double buffering. The transmit serial shifter cannot be directly accessed by the CPU. The output of the transmit serial shifter is connected to the TXD pin whenever the transmitter is operating (TE = 1, or TE = 0 and transmitter operation not yet complete).

The following definitions apply to the transmitter and receiver operation:

- **Bit-Time** — The time required to serially transmit or receive one bit of data, which is equal to one cycle of the baud frequency.
- **Start Bit** — One bit-time of logic zero that indicates the beginning of a data frame. A start bit must begin with a one-to-zero transition and be preceded by at least three receive time (RT) samples of logic one.
- **Stop Bit** — One bit-time of logic one that indicates the end of a data frame.
- **Frame** — A start bit, followed by a specified number of data or information bits, terminated by a stop bit. The number of data or information bits must agree between the transmitting and receiving devices. The most common frame format is one start bit followed by eight data bits (LSB first) terminated by one stop bit, for a total of 10 bit-times in the frame. The SCI optionally provides a 9-bit data format that results in an 11 bit-time frame.

The M bit in SCCR1 specifies the number of bit-times in the frame (ten or eleven). The most common format for non-return-to-zero (NRZ) serial interface is one start bit (logic zero or space), followed by eight data bits (transmitted LSB first), and one stop bit (logic one or mark). In addition to this standard format, the SCI provides hardware support for a 9-bit data format. This format is one start bit, eight data bits (LSB first), a parity or address (control) bit, and one stop bit. Following are all the possible formats:

- Start bit, seven data bits, two stop bits
- Start bit, seven data bits, address bit, one stop bit
- Start bit, seven data bits, address bit, two stop bits
- Start bit, seven data bits, parity bit, one stop bit
- Start bit, eight data bits, one stop bit
- Start bit, eight data bits, two stop bits
- Start bit, eight data bits, parity bit, one stop bit
- Start bit, eight data bits, address bit, one stop bit

When the transmitter is enabled by writing a one to TE in SCCR1, a check is made to determine if the transmit serial shifter is empty. If empty (TC = 1), a preamble consisting of all ones (no start bits) is transmitted. If the transmit serial shifter is not empty (TC = 0), then normal shifting continues until the word in progress with stop bit(s) is sent. The preamble (an all ones frame) is then transmitted.

When TE is cleared, the transmitter is disabled only after all pending information is transmitted, including any data in the transmit serial shifter (inclusive of the stop bit), any queued preamble (idle frame), or any queued break (logic zero frame). The TC flag is set, and the TXD pin reverts to control by PORTQS and DDRQS. This function allows the user to terminate a transmission sequence in the following manner. After

loading the last byte into register TDR and receiving the interrupt from TDRE in SCSR, (indicating that the data has transferred into the transmit serial shifter), the user clears TE. The last frame is transmitted normally, and the TXD pin reverts to control by PORTQS and DDRQS.

To insert a delimiter between two messages and place the nonlistening receivers in wakeup mode or to signal a retransmission (by forcing an idle line), TE is set to zero and then to one before the word in the transmit serial shifter has completed transmission. The transmitter waits until that word is transmitted and then starts transmission of a preamble (ten or eleven contiguous ones). After the preamble is transmitted, and if TDRE is set (no new data to transmit), the line continues to mark (remain high). Otherwise, normal transmission of the next word begins.

Two SCI messages may be separated with minimum idle time by using a preamble of ten bit-times (eleven if a 9-bit data format is specified) of marks (logic ones). The entire process can occur using the following procedure:

a. Write the last byte of the first message to the TDR.
b. Wait for TDRE to go high, indicating that the last byte is transferred to the transmit serial shifter.
c. Clear TE and then set TE back to one. This queues the preamble to follow the stop bit of the current transmission immediately.
d. Write the first byte of the second message to register TDR.

In this sequence, if the first byte of the second message is not transferred to register TDR prior to the finish of the preamble transmission, then the transmit data line (TXD pin) simply marks idle (logic one) until TDR is finally written. Also, if the last byte of the first message finishes shifting out (including the stop bit) and TE is clear, TC will go high and transmission will be considered complete. The TXD pin reverts to being a general-purpose I/O line.

The CPU writes data to be transmitted to register TDR, which automatically loads the data into the transmit serial shifter. Before writing to TDR, the user should check TDRE in SCSR. If TDRE = 0, then data is still waiting to be sent to the transmit serial shifter. Writing to TDR with TDRE clear overwrites previous data to be transferred. If TDRE = 1, then register TDR is empty and new data may be written to TDR clearing TDRE.

As soon as the data in the transmit serial shifter has shifted out and if a new byte of data is in TDR (TDRE = 0), then the new data is transferred from register TDR to the transmit serial shifter, and TDRE is automatically set. An interrupt may optionally be generated at this point.

The data in the transmit serial shifter is prefixed by a start bit (logic zero) and suffixed by the ninth data bit, if M = 1, and by one stop bit. The ninth data bit can be used as normal data or as an extra stop bit. A parity bit is substituted if PE = 1. This data stream is shifted out over the TXD pin. When the data is completely shifted out and no preamble or send break is requested, TC is set to one and the TXD pin remains high (logic one or mark).

Parity generation is enabled by setting PE in SCCR1 to a one. The last data bit, bit eight (or bit nine of the data if M = 1), is used as the parity bit, which is inserted between the normal data bits and the stop bit(s).

When TE is cleared, the transmitter yields control of the TXD pin in the following manner. If no information is being shifted out (i.e., if the transmitter is in an idle state, TC = 1), then the TXD pin reverts to being a general-purpose I/O pin. If a transmission is still in progress (TC = 0), the characters in the transmit serial shifter continue to be shifted out normally, followed by any queued break. When finished, TXD reverts to being a general-purpose I/O pin. To avoid terminating the transmitter before all data is transferred, the software should always wait for TDRE to be set before clearing TE.

Transmissions may be purposely aborted by the send break function. By writing SBK in SCCR1 to a one, a nonzero integer multiple of ten bit-times (eleven if 9-bit data format is specified) of space (logic zero) is transmitted. If SBK is set while a transmission is in progress, the character in the transmit serial shifter finishes normally (including the stop bit) before the break function begins. Break frames are sent until either SBK or TE is cleared. To guarantee the minimum break time, SBK should be quickly toggled to one and then back to zero. After the break time, at least one bit-time of mark idle (logic one) is transmitted to ensure that a subsequent start bit can be recognized.

The TXD pin has several control options to provide flexible operation. WOMS in SCCR1 can select either open-drain output (for wired-OR operation) or normal CMOS output. WOMS controls the function of the TXD pin whether the pin is being used for SCI transmissions (TE = 1) or as a general-purpose I/O pin.

In an SCI system with multiple transmitters, the wired-OR mode should be selected for the TXD pin of all transmitters, allowing multiple output pins to be coupled together. In the wired-OR mode, an external pull-up resistor on the TXD pin is necessary.

In some systems, a mark (logic one) signal is desired on the TXD pin, even when the transmitter is disabled. This is accomplished by writing a one to PORTQS in the appropriate position and configuring the TXD pin as an output in DDRQS. When the transmitter releases control of the TXD pin, it reverts to driving a logic one output, which is the same as mark or idle.

7.4.4 Receiver Operation

The receiver can be divided into two segments. The first is the receiver bit processor logic that synchronizes to the asynchronous receive data and evaluates the logic sense of each bit in the serial stream. The second receiver segment controls the functional operation and the interface to the CPU including the conversion of the serial data stream to parallel access by the CPU.

7.4.4.1 Receiver Bit Processor

The receiver bit processor contains logic to synchronize the bit-time of the incoming data and to evaluate the logic sense of each bit. To accomplish this an RT clock, which is 16 times the baud rate, is used to sample each bit. Each bit-time can thus be divided into 16 time periods called RT1–RT16. The receiver looks for a possible start bit by

watching for a high-to-low transition on the RXD pin and by assigning the RT time labels appropriately.

When the receiver is enabled by writing RE in SCCR1 to one, the receiver bit processor logic begins an asynchronous search for a start bit. The goal of this search is to gain synchronization with a frame. The bit-time synchronization is done at the beginning of each frame so that small differences in the baud rate of the receiver and transmitter are not cumulative. The SCI also synchronizes on all one-to-zero transitions in the serial data stream, which makes the SCI tolerant to small frequency variations in the received data stream.

The sequence of events used by the receiver to find a start bit is listed below.

a. Sample RXD input during each RT period and maintain these samples in a serial pipeline that is three RT periods deep.
b. If RXD is low during this RT period, go to step A.
c. If RXD is high during this RT period, store sample and proceed to step D.
d. If RXD is low during this RT period, but not high for the previous three RT periods (which is noise only), set an internal working noise flag and go to step A, since this transition was not a valid start bit transition.
e. If RXD is low during this RT period and has been high for the previous three RT periods, call this period RT1, set RAF, and proceed to step F.
f. Skip RT2 but place RT3 in the pipeline and proceed to step G.
g. Skip RT4 and sample RT5. If both RT3 and RT5 are high (RT1 was noise only), set an internal working noise flag. Go to step c and clear RAF. Otherwise, place RT5 in the pipeline and proceed to step H.
h. Skip RT6 and sample RT7. If any two of RT3, RT5, or RT7 is high (RT1 was noise only), set an internal working noise flag. Go to step c and clear RAF. Otherwise, place RT7 in the pipeline and proceed to step I.
i. A valid start bit is found and synchronization is achieved. From this point on until the end of the frame, the RT clock will increment starting over again with RT1 on each one-to-zero transition or each RT16. The beginning of a bit-time is thus defined as RT1 and the end of a bit-time as RT16.

Upon detection of a valid start bit, synchronization is established and is maintained through the reception of the last stop bit, after which the procedure starts all over again to search for a new valid start bit. During a frame's reception, the SCI resynchronizes the RT clock on any one-to-zero transitions.

Additional logic in the receiver bit processor determines the logic level of the received bit and implements an advanced noise-detection function. During each bit-time of a frame (including the start and stop bits), three logic-sense samples are taken at RT8, RT9, and RT10. The logic sense of the bit-time is decided by a majority vote of these three samples. This logic level is shifted into register RDR for every bit except the start and stop bits.

If RT8, RT9, and RT10 do not all agree, an internal working noise flag is set. Additionally for the start bit, if RT3, RT5, and RT7 do not all agree, the internal working noise flag is set. If this flag is set for any of the bit-times in a frame, the NF flag in SCSR is set concurrently with the RDRF flag in SCSR when the data is transferred to register RDR. The user must determine if the data received with NF set is valid. Noise on the RXD pin does not necessarily corrupt all data.

The operation of the receiver bit processor is shown in the following figures. These examples demonstrate the search for a valid start bit and the synchronization procedure as outlined above. The possibility of noise durations greater than one bit-time are not considered in these examples. Figure 7-14 illustrates the ideal case with no noise present.

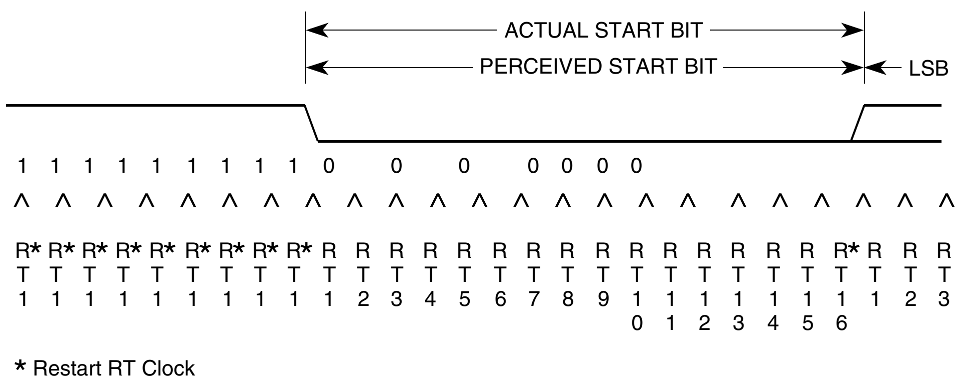


Figure 7-14 Start Search Example 1

Figure 7-15 shows the start bit search and resynchronization process being restarted because the first low detected was determined to be noise rather than the beginning of a start bit-time. Since the noise occurred before the start bit was found, it will not cause the internal working noise flag to be set.

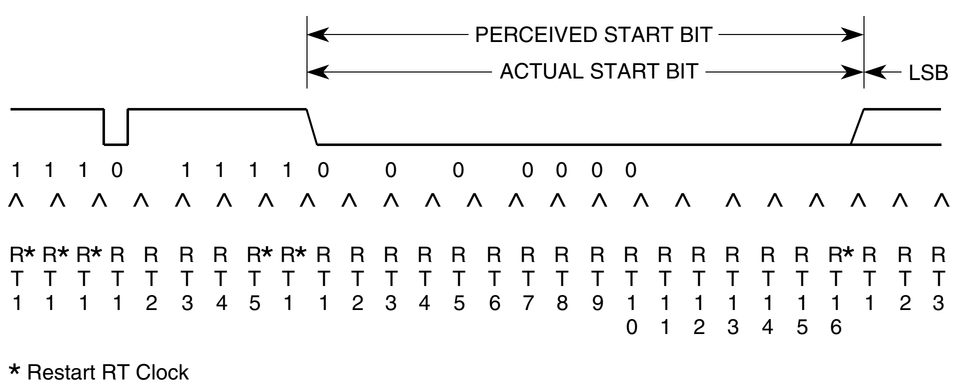


Figure 7-15 Start Search Example 2

Figure 7-16 shows that noise is perceived as the beginning of a start bit. Note that the high level sensed at RT3 causes the internal working noise flag to be set. Even though this figure shows improper alignment of the perceived bit-time boundaries to the actual bit-time boundaries, the logic sense samples taken at RT8, RT9, and RT10 fall well within the correct actual bit-time. The start bit and all other bits in the frame should be received correctly.

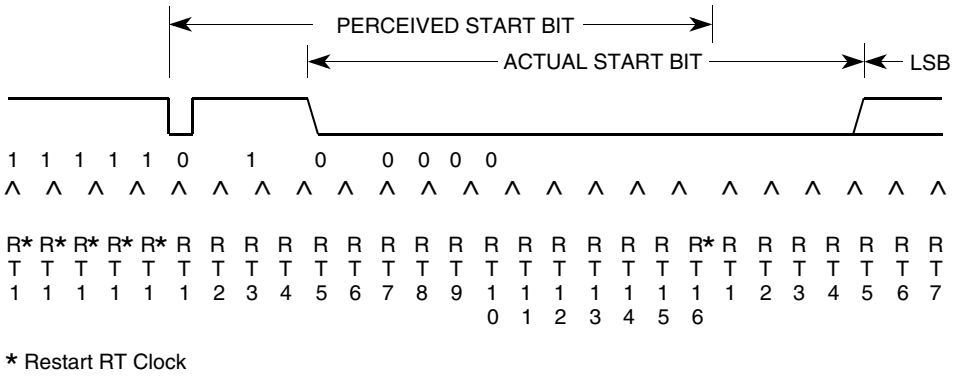


Figure 7-16 Start Search Example 3

Figure 7-17 shows how a large burst of noise is perceived as the beginning of a start bit. Note that RT5 is sensed logic high, setting the internal working noise flag. This figure also illustrates a worst-case alignment of the perceived bit-time boundaries to the actual bit-time boundaries; however, RT8, RT9, and RT10 all fall within the correct actual bit-time. The start bit is detected and the incoming data stream is correctly sensed.

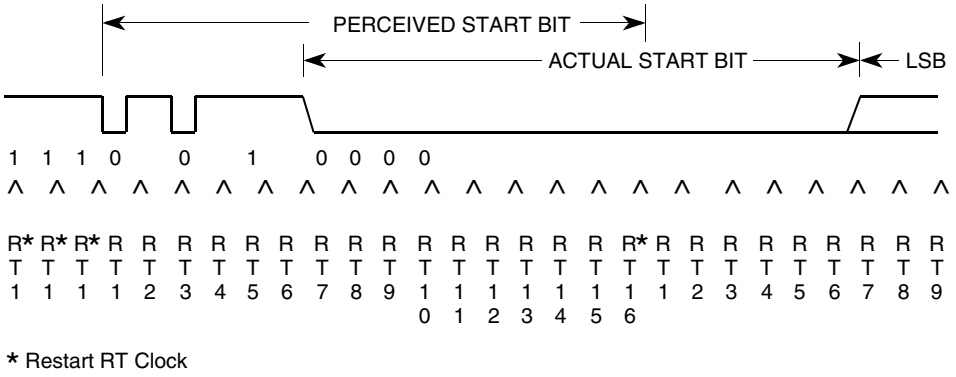


Figure 7-17 Start Search Example 4

Figure 7-18 illustrates the effect of noise early within the start bit-time. Although this noise does not affect proper synchronization with the start bit-time, it does set the internal working noise flag.

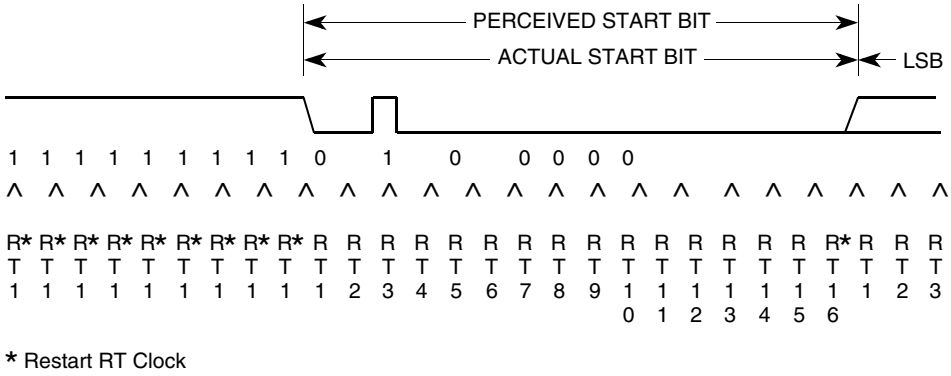


Figure 7-18 Start Search Example 5

Figure 7-19 shows a large burst of noise near the beginning of the start bit that causes the start bit search to be restarted. During RT1 following RT7, a search for a new start bit could not be started as the previous three RT samples are not all high. The receiver bit processor misses this start bit. The frame might be partially received or missed entirely, depending on the data in the frame and when the start bit search logic synchronized upon what appeared to be a start bit. If a valid stop bit is not detected, an FE flag is set in SCSR.

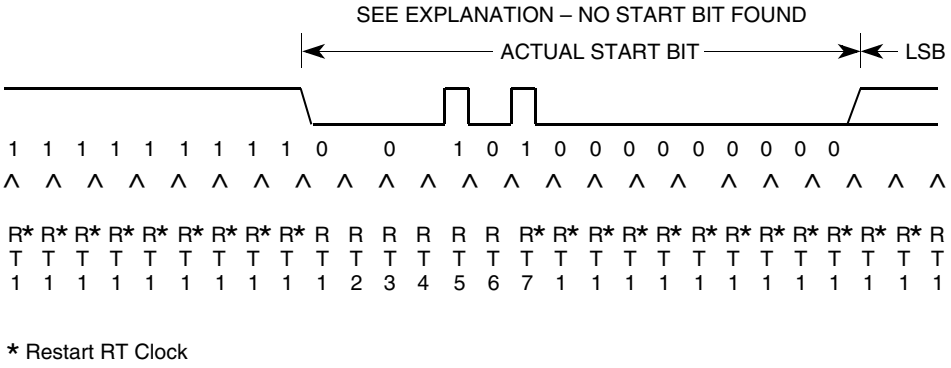


Figure 7-19 Start Search Example 6

Figure 7-20 explores the case where the majority vote of RT8, RT9, and RT10 returns a logic-high level. However, the start bit is a special case that overrules the majority voting scheme. In review, at least three of the samples taken at RT1, RT3, RT5, and RT7 must be low. The start bit is detected and the RT clock is synchronized; because RT8–RT10 were not unanimous, the NF flag is set.

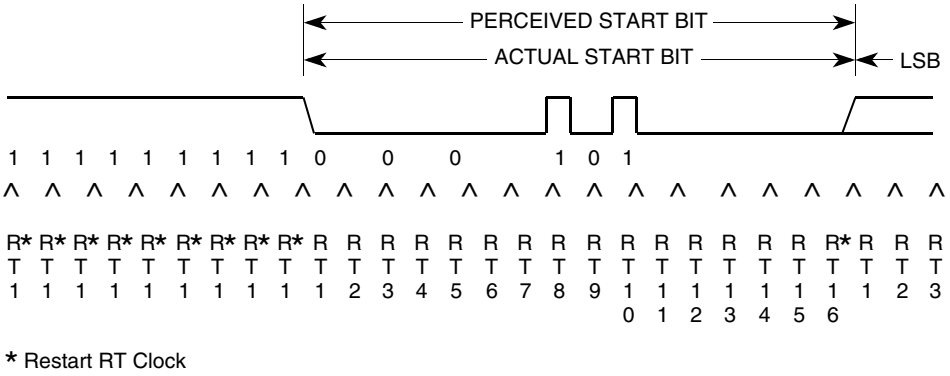


Figure 7-20 Start Search Example 7

7.4.4.2 Receiver Functional Operation

The receiver contains a receive serial shifter and a parallel RDR. While one character is in the process of being shifted in, another character may be held in RDR. This capability is called double buffering. The receive serial shifter cannot be accessed directly by the CPU. The input of the receive serial shifter is connected to the majority sampling logic of the receive bit processor.

The receiver is enabled when RE in SCCR1 is set to one. When RE is zero, the receiver is initialized and most of the receiver bit processor logic is disabled. The receiver bit processor logic drives a state machine (run by the RT clock) that determines the logic level for each bit-time. This state machine controls when the bit processor logic is to sample the RXD pin and also controls when data is to be passed to the receive serial shifter. Data is shifted into the receive serial shifter according to the most recent synchronization of the RT clock with the incoming data stream. From this point on, the data is moved synchronously with the MCU system clock.

The first bit shifted in is the start bit, which is always a logic zero. The next eight bits shifted in are the basic data byte (LSB first). The next bit shifted in depends on the mode selected by M in SCCR1. If M = 1, then the bit is the ninth data bit and is placed in R8 of SCDR, concurrent with the transfer of data from the receive serial shifter to register RDR.

The last bit shifted in for each frame is the stop bit, which is always a logic one. If a logic zero is sensed during this bit-time, the FE error flag in SCSR is set. A framing error is usually caused by mismatched baud rates between the receiver and transmitter or by a significant burst of noise. Note that a framing error is not always caught; the data in the expected stop bit-time may be a logic one regardless.

When the stop bit is received, the frame is considered to be complete, and the received character in the receive serial shifter is transferred in parallel to RDR. If M = 1, the ninth bit is transferred at the same time; however, if the RDRF flag in SCSR is set, transfers are inhibited. Instead, the OR error flag is set, indicating to the user that

the CPU needs to service register RDR faster. The data in RDR is preserved, but the data in the receive serial shifter is lost.

All status flags associated with a serially received frame are set simultaneously and at a time that does not interfere with CPU access to the affected registers. When a completed frame is received, either the RDRF or OR flag is always set. If RIE in SCCR1 is set, an interrupt results whenever RDRF is set. The receiver status flags NF, FE, and PF are set simultaneously with RDRF, as appropriate. These receiver flags are never set with OR because the flags only apply to the data in the receive serial shifter. The receiver status flags do not have separate interrupt enables, since they are set simultaneously with RDRF and must be read by the user at the same time as RDRF.

All receiver status flags are cleared by the following sequence. Register SCSR is read first, followed by a read of register SCDR. Reading SCSR not only informs the CPU of the status of the received data, but also arms the clearing mechanism. Reading SCDR supplies the received data to the CPU and clears all of the status flags: RDRF, IDLE, OR, NF, FE, and PF.

7.4.4.3 Idle-Line Detect

The receiver hardware includes the ability to detect an idle line. This function can be used to indicate when a group of serial transmissions is finished. An idle line is defined as a minimum of ten bit-times (or eleven if a 9-bit data format is selected) of contiguous ones on the RXD pin. During a typical serial transmission, frames are transmitted isochronously, that is, no idle time occurs between frames. Even if all data bits in a frame are logic ones, the start bit ensures that at least one logic zero bit-time occurs for each frame.

Motorola MCUs from the M68HC11 and M68HC05 families have SCIs with only one type of idle-line detect circuitry. On these MCUs, the receiver bit processor starts counting logic one bit-times at any point (even within a frame). This method allows the earliest recognition of an idle line because the stop bit and any contiguous ones preceding the stop bit are counted with the logic ones in the idle line following the stop bit. In some applications, the CPU overhead prevents the servicing of interrupts as soon as possible to ensure that no bit-time of an idle line occurs between frames. Although this idle line causes no deterioration of the message content, if one bit-time should occur after a data byte of all ones, the combination is seen as an idle line and causes sleeping SCIs to wake up.

The SCI on the QSM module contains this same idle-line detect logic called short idle-line detect as well as long idle-line detect. In long idle-line detect mode, the SCI begins counting logic ones after the stop bit is received. The data content of a byte, therefore, does not affect how quickly the idle line is detected. When RXD goes idle for the minimum required time, the IDLE flag in SCSR is set. ILT in SCCR1 is used to choose between short and long idle-line detection.

If ILIE in SCCR1 is set, a hardware interrupt request is generated when the IDLE flag is set. This flag is cleared by reading SCSR with IDLE set, followed by reading register RDR. The IDLE flag is not set again until after at least one frame has been received

(RDRF = 1), which prevents an extended idle interval from causing more than one interrupt.

7.4.4.4 Receiver Wakeup

The SCI receiver hardware provides a receiver wakeup function to support multinode networks containing more than one receiver. This function allows the transmitting device to direct a message to an individual receiver or group of receivers by sending an address frame at the start of a message. All receivers not addressed for the current message invoke the receiver wakeup function, which effectively allows them to sleep through the rest of the message. Therefore, the CPU is alleviated from servicing register RDR, resulting in increased system performance.

The SCI receiver is placed in wakeup mode by writing a one to RWU in SCCR1. While RWU is set, all receiver status flag bits are inhibited from being set. Note that the IDLE flag cannot be used when RWU is set. Although the CPU can clear RWU by writing a zero to SCCR1, it is normally left alone by software and is cleared automatically by hardware in one of two methods: idle-line wakeup or address-mark wakeup.

WAKE in SCCR1 determines which method of wakeup is to be employed. If WAKE = 0, idle-line wakeup is selected. This method is compatible with the method originally used on the MC6801. If WAKE = 1, address-mark wakeup is selected, which uses a one in the MSB of data to denote an address frame and uses a zero to denote a normal data frame. Each method has its particular advantages and disadvantages.

Both wakeup methods require a software device addressing and recognition scheme and, therefore, can conform to all transmitters and receivers. The addressing information is usually the first frame(s) of the message. Receivers for which the message is not intended may set RWU and go back to sleep for the remainder of the message.

Idle-line wakeup allows a receiver to sleep until an idle line is detected, causing RWU to be cleared by the receiver and causing the receiver to wake up. The receiver waits through the idle times for the first frame of the next message. If the receiver is not the intended addressee, RWU may be set to put the receiver back to sleep. This method of receiver wakeup requires that a minimum of one frame of idle line be imposed between messages. As previously stated, no idle time is allowed between frames within a message.

Address-mark wakeup uses a special frame format to wake up the receiver. All frames consist of seven (or eight) data bits plus an MSB that indicates an address frame when set to a one. The first frame of each message should be an address frame. All receivers in the system must use a software scheme to determine which messages address them. If the message is not intended for a particular receiver, the CPU sets RWU so that the receiver goes back to sleep, thereby eliminating additional CPU overhead for servicing the rest of the message.

When the first frame of a new message is received with the MSB set, denoting an address frame, RWU is cleared. The byte is received normally, transferred to register RDR, and the RDRF flag is set. Address-mark wakeup allows messages to include idle

times between frames and eliminates idle time between messages; however, an efficiency loss results from the extra bit-time (address bit) that is required on all frames.

7.5 Bus Error Support

Bus error support has been added to the QSM. When the QSM is configured for bus error assertion, the QSM will generate a bus error for reserved register accesses and privilege violations. Specifically, bus errors are generated for the following:

- Access to a reserved 16-bit register within the QSM's memory map.
- User access to supervisor-only registers when other registers in the QSM's memory map are user-accessible (QMCR SUPV bit is not set).
- Access to the test register (QTEST) when not in test mode.
- Writes to read-only registers.

The QMCR, QTEST, QILR, and QIVR registers are always supervisor-only accessible registers. The only register that is read-only is the SCI's status register (SCSR). A write to this register generates a bus error. The reserved registers are at offsets 0x06, 0x10, 0x12, 0x20-0xFF, and 0x150-0x1FF to the QSM base address.

If the QSM is not configured for bus error assertion then the BIU does not generate bus errors. For all of the above conditions the BIU asserts DTACK to terminate the cycle. Writes have no effect and reads return zero data.

7.6 Interrupt Request Functionality

The QSM supports two different schemes for asserting interrupts. The QSM can be factory configured to support bus masters with interrupt acknowledge (IACK) cycles and vector generation or configured for bus masters with interrupt level byte selects (ILBS) cycles. The features unique to each will be discussed separately.

7.6.1 Interrupts for Bus Masters that Support IACK

For bus masters that support IACK cycles (and not ILBS) with vector generation the QSM supports seven levels of interrupt priority. The level generated is specified in the QSM interrupt level register (QILR) for both the QSPI and the SCI (refer to **7.2.2.2 QSM Interrupt Level Register (QILR)** for details). A level seven interrupt specifies the highest level of priority, while a level one interrupt specifies the lowest level of priority. Specifying a level zero interrupt for either the QSPI or SCI disable interrupts for that sub-module.

After the QSPI or SCI issues an interrupt request and the bus master recognizes it, an IACK cycle is issued by the bus master. During the IACK cycle all modules in the system currently requesting interrupt service for the level specified by the IACK cycle will respond to the cycle. The QSM compares the pending levels in QILR with the IACK level to determine if a response is required.

In the case of multiple modules requesting interrupt service for the same interrupt level, the interrupt arbitration (IARB) field of each module will break the tie. The module with the highest IARB value wins arbitration. For the QSM the IARB value is specified in register QMCR. An IARB value of 0x0 disables arbitration.

If the QSM is the only module requesting interrupt service, or it wins during the ensuing arbitration, it responds to the IACK cycle with an 8-bit interrupt vector number. The CPU uses this vector as a displacement into the exception vector table, then uses the resulting vector to jump to an interrupt service routine. The vector number used by the QSM is specified in register QIVR.

7.6.2 Pad Driver Level Modifications

Eight of the nine QSM pins can be configured by the user as general-purpose I/O ports. These pins are PCS3, PCS2, PCS1, PCS0/SS, SCK, MOSI, MISO, and TXD. In order to reduce electro-magnetic interference (EMI) on signal transitions of this user assignable port, eight new pad control signals have been added.

Each pad control signal is connected to the input/output pad of its corresponding QSM pin. A high logic level on a pin assignment pin means the corresponding data pin is being used by the QSPI or SCI and the I/O pad is configured for fast signal transition delay. A low logic level means the data pin is being used as a general-purpose I/O and the I/O pad is configured for slow signal transition delay.

These logic levels match the state of the of the pin assignment bits in the QSM pin assignment register (QPAR), except for SCK and TXD. SCK and TXD are only general-purpose ports when the QSPI or SCI, respectively, are disabled.

Each pad control signal is connected to the input/output pad of its corresponding QSM pin. If a pin is configured as an I/O pin, the I/O pad is configured for slow transition delay. If a pin is being used by the QSPI or SCI, the I/O pad is configured for fast transition delay.

SECTION 8
QUEUED ANALOG-TO-DIGITAL CONVERTER MODULE-64

MC68377 includes one independent queued analog-to-digital converter (QADC64) module.

8.1 Overview

The QADC64 modules consist of an analog front-end and a digital control subsystem, which includes an intermodule bus (IMB3) interface block. Refer to **Figure 8-1**.

The analog section includes input pins, channel selection logic, an analog multiplexer, and one sample and hold analog circuit. The analog conversion is performed by the digital-to-analog converter (DAC) resistor-capacitor array, a high-gain comparator, and a successive approximation register (SAR).

The digital control section contains the conversion sequencing logic. Also included are the periodic/interval timer, control and status registers, the conversion command word (CCW) table RAM, and the result word table RAM.

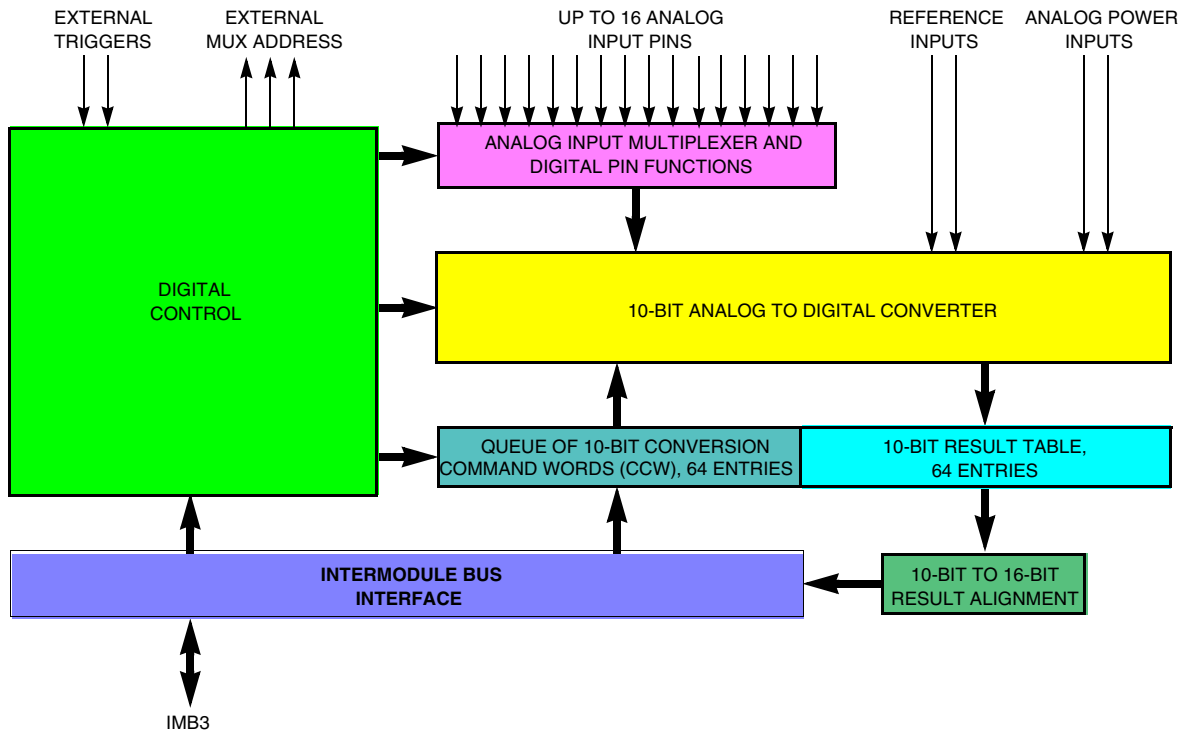


Figure 8-1 QADC64 Block Diagram

8.2 Features

The QADC64 module offers the following features:

- Internal sample and hold
- Up to 16 analog input channels using internal multiplexing
- Directly supports up to four external multiplexers (for example, the MC14051)
- Up to 41 total input channels with internal and external multiplexing
- Programmable input sample time for various source impedances
- Two conversion command queues with a total of 64 entries
- Sub-queues possible using pause mechanism
- Queue complete and pause software interrupts available on both queues
- Queue pointers indicate current location for each queue
- Automated queue modes initiated by:
  - — External edge trigger [queues 1 and 2] and gated mode [queue 1 only]
  - — Periodic/interval timer, within QADC64 module [queues 1 and 2]
  - — Software command [queues 1 and 2]
- Single-scan or continuous-scan of queues
- 64 result registers
- Output data readable in three formats:
  - — Right-justified unsigned
  - — Left-justified signed
  - — Left-justified unsigned
- Unused analog channels can be used as digital ports

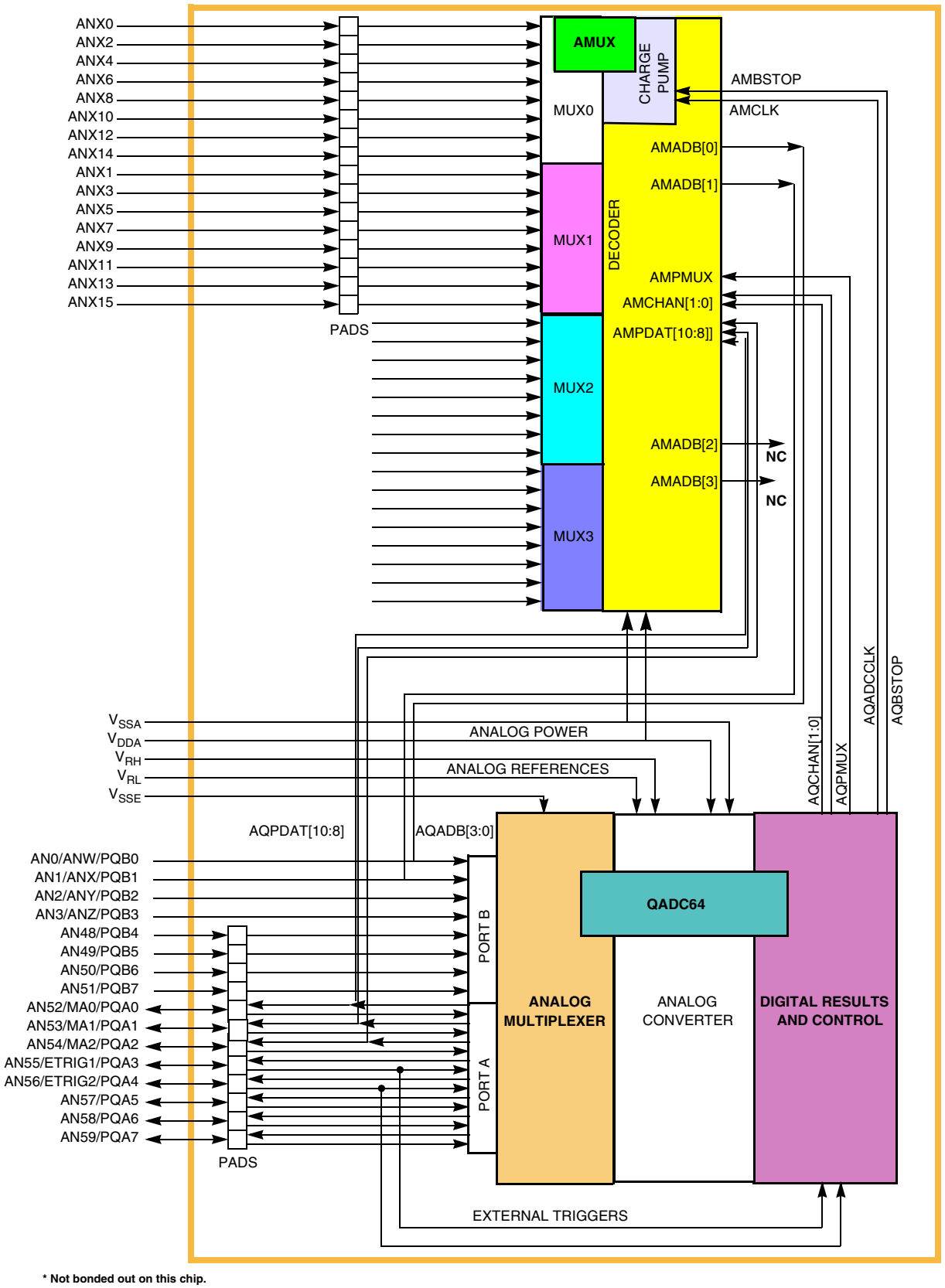
8.3 QADC64 Pin Functions

The QADC64 module uses the following 36 pins:

- Two analog reference pins, to which all analog input voltages are scaled
- 16 analog input pins with three analog inputs multiplexed with multiplex address signals and two multiplexed with the on-chip analog multiplexer inputs
- 16 analog input pins through the on-chip AMUX circuit
- Two analog power pins (V_{DDA}, V_{SSA})
- Two trigger pins shared with AN[55:56]/PQA[3:4]

The 16 channel/port pins can support up to 41 channels when external multiplexing is used (including internal channels). All of the channel pins can also be used as general-purpose digital port pins.

The following paragraphs describe QADC64 pin functions. **Figure 8-2** shows the QADC64 module pins.



* Not bonded out on this chip.

Figure 8-2 QADC64 Input and Output Signals

8.3.1 Port A Pin Functions

The eight port A pins can be used as analog inputs or as a bidirectional 8-bit digital input/output port.

8.3.1.1 Port A Analog Input Pins

When used as analog inputs, the eight port A pins are referred to as AN[59:52]. Due to the digital output drivers associated with port A, the analog characteristics of port A are different from those of port B. All of the analog signal input pins may be used for at least one other purpose.

8.3.1.2 Port A Digital Input/Output Pins

Port A pins are referred to as PQA when used as a bidirectional 8-bit digital input/output port. These eight pins may be used for general-purpose digital input signals or digital output signals.

Port A pins are connected to a digital input synchronizer during reads and may be used as general purpose digital inputs. Since port A read captures the data on all pins, including those used for digital outputs or analog inputs, the user should employ a “masking” operation to filter the inappropriate bits from the input byte.

Each port A pin is configured as input or output by programming the port data direction register (DDRQA). Digital input signal states are read into the PORTQA data register when DDRQA specifies that the pins are inputs. Digital data in PORTQA is driven onto the port A pins when the corresponding bits in DDRQA specify outputs.

8.3.2 Port B Pin Functions

The eight port B pins can be used as analog inputs, or as an 8-bit digital input only port. Refer to the following paragraphs for more information.

8.3.2.1 Port B Analog Input Pins

When used as analog inputs, the eight port B pins are referred to as AN[51:48]/AN[3:0]. Since port B functions as analog and digital input only, the analog characteristics are different from those of port A. All of the analog signal input pins may be used for at least one other purpose.

8.3.2.2 Port B Digital Input Pins

Port B pins are referred to as PQB[7:0] when used as an 8-bit digital input only port. In addition to functioning as analog input pins, the port B pins are also connected to the input of a synchronizer during reads and may be used as general-purpose digital inputs.

Since port B pins are input only, there is no associated data direction register. Digital input signal states are read from the PORTQB data register. Since a port B read captures the data on all pins, including those used for analog inputs, the user should employ a “masking” operation to filter the inappropriate bits from the input byte.

8.3.3 External Trigger Input Pins

The QADC64 has two external trigger pins (ETRIG[2:1]). Each of the two external trigger pins is associated with one of the scan queues. When a queue is in external trigger mode, the corresponding external trigger pin is configured as a digital input.

8.3.4 Multiplexed Address Output Pins

In non-multiplexed mode, the 16 channel pins are connected to an internal multiplexer which routes the analog signals into the A/D converter.

In externally multiplexed mode, the QADC64 allows automatic channel selection through up to four external 1-of-8 multiplexer chips. The QADC64 provides a 3-bit multiplexed address output to the external multiplexer chips to allow selection of one of eight inputs. The multiplexed address output signals MA[2:0] can be used as multiplex address output bits or as general-purpose I/O.

When externally multiplexed mode is enabled, MA[2:0] are used as the address inputs for up to four 1-of-8 multiplexer chips (for example, the MC14051 and the MC74HC4051). Since MA[2:0] are digital outputs in multiplexed mode, the software programmed input/output direction and data for these pins in DDQA[2:0], DDRQA, and PQA[2:0] is ignored, and the value for MA[2:0] is taken from the currently executing CCW.

8.3.5 Multiplexed Analog Input Pins

In externally multiplexed mode, four of the port B pins are redefined to each represent a group of eight input channels. Refer to **Table 8-1**.

The analog output of each external multiplexer chip is connected to one of the AN[w, x, y, z] inputs in order to convert a channel selected by the MA[2:0] multiplexed address outputs.

Table 8-1 Multiplexed Analog Input Channels

Multiplexed Analog Input	Channels
ANw[1]	Even numbered channels from 0 to 14
ANx[1]	Odd numbered channels from 1 to 15
ANy[2]	Even numbered channels from 16 to 30
ANz[2]	Odd numbered channels from 17 to 31

NOTES:
1. If the on-chip multiplexer is enabled, ANw and ANx are used as inputs for the AMUX outputs.
2. If the on-chip AMUX is enabled, then AN2 and AN3 should be read as channels AN16 and AN17.

8.3.6 Voltage Reference Pins

V_{RH} and V_{RL} are the dedicated input pins for the high and low reference voltages. Separating the reference inputs from the power supply pins allows for additional external

filtering, which increases reference voltage precision and stability, and subsequently contributes to a higher degree of conversion accuracy.

8.3.7 Dedicated Analog Supply Pins

V_{DDA} and V_{SSA} pins supply power to the analog subsystems of the QADC64 module. Dedicated power is required to isolate the sensitive analog circuitry from the normal levels of noise present on the digital power supply.

8.3.8 External Digital Supply Pin

Each port A pin includes a digital output driver, an analog input signal path, and a digital input synchronizer. The V_{SSE} pin provides the ground level for the drivers on the port A pins. V_{DDH} provides the supply level for the drivers on port A pins.

8.3.9 Digital Supply Pins

V_{DD} and V_{SS} provide the power for the digital portions of the QADC64, and for all other digital MCU modules.

8.4 QADC64 Bus Interface

The QADC64 supports 8-bit, 16-bit, and 32-bit data transfers at even and odd addresses. Coherency of results read (ensuring that all results read were taken consecutively in one scan) is not guaranteed. For example, if two consecutive 16-bit locations in a result area are read, the QADC64 could change one 16-bit location in the result area between bus cycles. There is no holding register for the second 16-bit location. All read and write accesses that require more than one 16-bit access to complete occur as two or more independent bus cycles. Depending on bus master protocol, these accesses could include misaligned and 32-bit accesses.

Normal reads from and writes to the QADC64 require two clock cycles. However, if the CPU tries to access locations that are also accessible to the QADC64 while the QADC64 is accessing them, the bus cycle will require additional clock cycles. The QADC64 may insert from one-to-four wait states in the process of a CPU read from, or write to, such a location.

8.5 Module Configuration

The QADC64 module configuration register (QADC64MCR) defines freeze and stop mode operation, supervisor space access, and interrupt arbitration priority. Unimplemented bits read zero and writes have no effect. QADC64MCR is typically written once when software initializes the QADC64, and not changed thereafter. Refer to **8.12.1 QADC64 Module Configuration Register** for register and bit descriptions.

8.5.1 Low-Power Stop Mode

When the STOP bit in QADC64MCR is set, the clock signal to the A/D converter is disabled, effectively turning off the analog circuitry. This results in a static, low power consumption, idle condition. Low-power stop mode aborts any conversion sequence in progress. Because the bias currents to the analog circuits are turned off in low-

power stop mode, the QADC64 requires some recovery time (t_{SR}, see **APPENDIX E ELECTRICAL AND AC CHARACTERISTICS**) to stabilize the analog circuits after the STOP bit is cleared.

In low-power stop mode, the BIU state machine and logic do not shut down: the QADC64MCR and the interrupt register (QADC64INT) are fully accessible and are not reset. The data direction register (DDRQA), port data register (PORTQA/PORTQB), and control register 0 (QACR0) are not reset and are read-only accessible. The RAM is not reset and is not accessible. Control register 1 (QACR1), control register 2 (QACR2), and the status registers (QASR0 and QASR1) are reset and are read-only accessible. In addition, the periodic/interval timer is held in reset during stop mode.

If the STOP bit is clear, low-power stop mode is disabled. The STOP bit must be clear to program CCW's into RAM or read results from RAM.

8.5.2 Freeze Mode

The QADC64 enters freeze mode when background debug mode is enabled and a breakpoint is processed. This is indicated by assertion of the FREEZE line on the IMB3. The FRZ bit in QADC64MCR determines whether or not the QADC64 responds to an IMB FREEZE assertion. Freeze mode is useful when debugging an application.

When the IMB FREEZE line is asserted and the FRZ bit is set, the QADC64 finishes any conversion in progress and then freezes. Depending on when the FREEZE is asserted, there are three possible queue freeze scenarios:

- When a queue is not executing, the QADC64 freezes immediately.
- When a queue is executing, the QADC64 completes the current conversion and then freezes.
- If during the execution of the current conversion, the queue operating mode for the active queue is changed, or a queue 2 abort occurs, the QADC64 freezes immediately.

When the QADC64 enters the freeze mode while a queue is active, the current CCW location of the queue pointer is saved.

During freeze, the analog clock, QCLK, is held in reset and the periodic/interval timer is held in reset. External trigger events that occur during the freeze mode are not captured. The BIU remains active to allow IMB access to all QADC64 registers and RAM. Although the QADC64 saves a pointer to the next CCW in the current queue, the software can force the QADC64 to execute a different CCW by writing new queue operating modes for normal operation. The QADC64 looks at the queue operating modes, the current queue pointer, and any pending trigger events to decide which CCW to execute.

If the FRZ bit is clear, assertion of the IMB FREEZE line is ignored.

8.5.3 Supervisor/Unrestricted Address Space

The QADC64 memory map is divided into two segments: supervisor-only data space and assignable data space. Access to supervisor-only data space is permitted only

when the CPU is operating in supervisor mode. Assignable data space can have either restricted to supervisor-only data space access or unrestricted supervisor and user data space accesses. The SUPV bit in QADC64MCR designates the assignable space as supervisor or unrestricted.

Attempts to read or write supervisor-only data space when the CPU is not in supervisor mode cause the bus master to assert the internal transfer error acknowledge ($\overline{TEA}$) signal.

The supervisor-only data space segment contains the QADC64 global registers, which include QADC64MCR and QADC64INT. The supervisor/unrestricted space designation for the CCW table, the result word table, and the remaining QADC64 registers is programmable.

8.6 General-Purpose I/O Port Operation

QADC64 port pins, when used as general-purpose input, are conditioned by a synchronizer with an enable feature. The synchronizer is not enabled until the QADC64 decodes an IMB bus cycle which addresses the port data register to minimize the high-current effect of mid-level signals on the inputs used for analog signals. Digital input signals must meet the input low voltage (V_{IL}) or input high voltage (V_{IH}) requirements, see **APPENDIX E ELECTRICAL AND AC CHARACTERISTICS**. If an analog input pin does not meet the digital input pin specifications when a digital port read operation occurs, an indeterminate state is read. To avoid reading inappropriate values on analog inputs, the user software should employ a “masking” operation.

During a port data register read, the actual value of the pin is reported when its corresponding bit in the data direction register defines the pin to be an input (port A only). When the data direction bit specifies the pin to be an output, the content of the port data register is read. By reading the latch which drives the output pin, read-modify-write software instructions (like bit manipulation instructions).

There is one special case to consider for digital I/O port operation. When the MUX (externally multiplexed) bit is set in QACR0, the data direction register settings are ignored for the bits corresponding to PQA[2:0] and the three multiplexed address MA[2:0] output pins. The MA[2:0] pins are forced to be digital outputs, regardless of the data direction setting, and the multiplexed address outputs are driven. The data returned during a port data register read is the value of the multiplexed address latches which drive MA[2:0], regardless of the data direction setting.

8.6.1 Port Data Register

QADC64 ports A and B are accessed through two 8-bit port data registers (PORTQA and PORTQB). Port A pins are referred to as PQA when used as an 8-bit input/output port. Port A can also be used for analog inputs AN[59:52] and external multiplexer address outputs MA[2:0].

Port B pins are referred to as PQB when used as an 8-bit input-only digital port. Port B can also be used for non-multiplexed AN[51:48]/AN[3:0] and multiplexed ANz, ANy, ANx, ANw analog inputs.

PORTQA and PORTQB are unaffected by reset. Refer to **8.12.3 Port A/B Data Register** for register and bit descriptions.

8.6.2 Port Data Direction Register

The port data direction register (DDRQA) is associated with the port A digital I/O pins. These bidirectional pins may have somewhat higher leakage and capacitance specifications. Refer to **APPENDIX E ELECTRICAL AND AC CHARACTERISTICS** for more information.

Any bit in this register set to one configures the corresponding pin as an output. Any bit in this register cleared to zero configures the corresponding pin as an input. Software is responsible for ensuring that DDRQA bits are not set to one on pins used for analog inputs. When a DDRQA bit is set to one and the pin is selected for analog conversion, the voltage sampled is that of the output digital driver as influenced by the load.

NOTE

Caution should be exercised when mixing digital and analog inputs. This should be minimized as much as possible. Input pin rise and fall times should be as large as possible to minimize AC coupling effects.

Since port B is input-only, a data direction register is not needed. Read operations on the reserved bits in DDRQA return zeros, and writes have no effect. Refer to **8.12.4 Port Data Direction Register** for register and bit descriptions.

8.7 External Multiplexing Operation

External multiplexers concentrate a number of analog signals onto a few inputs to the analog converter. This is helpful in applications that need to convert more analog signals than the A/D converter can normally provide. External multiplexing also puts the multiplexer closer to the signal source. This minimizes the number of analog signals that need to be shielded due to the close proximity of noisy, high speed digital signals near the MCU.

NOTE

The main QADC64 treats the AMX as an external multiplexer. It is recommended that full external multiplexing not be used on MC68377. Mixed internal and external multiplexing should be used. See **8.13.2 Mixed AMUX/External Multiplexing** and **Figure 8-13**.

The QADC64 can use from one-to-four external multiplexers to expand the number of analog signals that may be converted. Up to 32 analog channels can be converted through external multiplexer selection. The externally multiplexed channels are automatically selected from the channel field of the conversion command word (CCW) table, the same as internally multiplexed channels.

All of the automatic queue features are available for externally and internally multiplexed channels. The software selects externally multiplexed mode by setting the MUX bit in QACR0.

Figure 8-3 shows the maximum configuration of four external multiplexers connected to the QADC64. The external multiplexers select one of eight analog inputs and connect it to one analog output, which becomes an input to the QADC64. The QADC64 provides three multiplexed address signals (MA[2:0]), to select one of eight inputs. These outputs are connected to all four multiplexers. The analog output of each multiplexer is connected to one of four separate QADC64 inputs — ANw, ANx, ANy, and ANz.

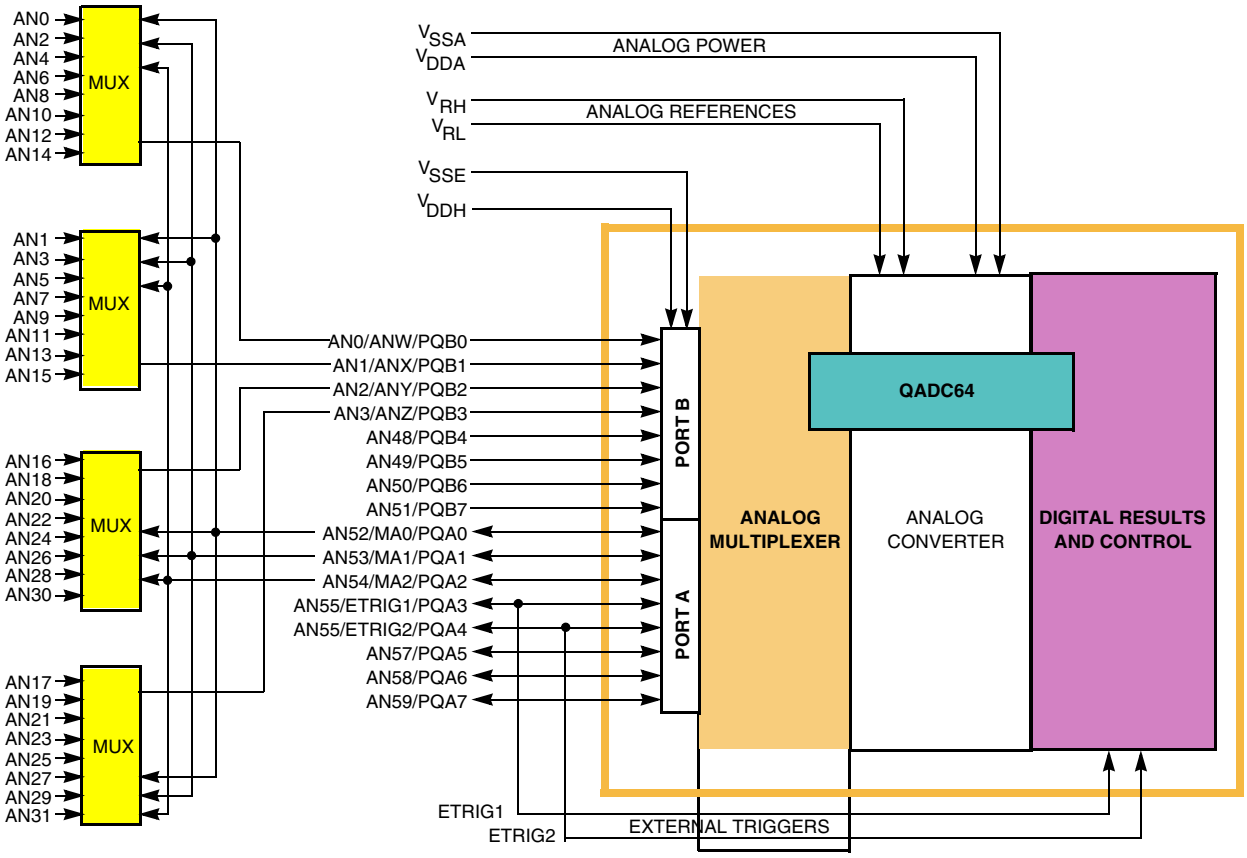


Figure 8-3 Example of Full External Multiplexing

When the external multiplexed mode is selected, the QADC64 automatically creates the MA[2:0] output signals from the channel number in each CCW. The QADC64 also converts the proper input channel (ANw, ANx, ANy, and ANz) by interpreting the CCW channel number. As a result, up to 32 externally multiplexed channels appear to the conversion queues as directly connected signals. Software simply puts the channel number of an externally multiplexed channel into a CCW.

Figure 8-3 shows that MA[2:0] may also be analog or digital input pins. When external multiplexing is selected, none of the MA[2:0] pins can be used for analog or digital inputs. They become multiplexed address outputs.

8.8 Analog Input Channels

The number of available analog channels varies, depending on whether or not external multiplexing is used. A maximum of 16 analog channels are supported by the internal multiplexing circuitry of the converter. Table 8-2 shows the total number of analog input channels supported with zero-to-four external multiplexers.

Table 8-2 Analog Input Channels

Number of Analog Input Channels Available Directly Connected + External Multiplexed = Total Channels[1]				
No External Mux Chips	**One External Mux Chip**	**Two External Mux Chips**	**Three External Mux Chips**	**Four External Mux Chips**
16	12 + 8 = 20	11 + 16 = 27	10 + 24 = 34	9 + 32 = 41

NOTES:
1. When external multiplexing is used, three input channels become multiplexed address outputs, and for each external multiplexer chip, one input channel becomes a multiplexed analog input.

8.9 Analog Subsystem

The QADC64 analog subsystem includes a front-end analog multiplexer, a digital-to-analog converter (DAC) array, a comparator, and a successive approximation register (SAR).

The analog subsystem path runs from the input pins through the input multiplexing circuitry, into the DAC array, and through the analog comparator. The output of the comparator feeds into the SAR.

Figure 8-4 shows a block diagram of the QADC64 analog submodule.

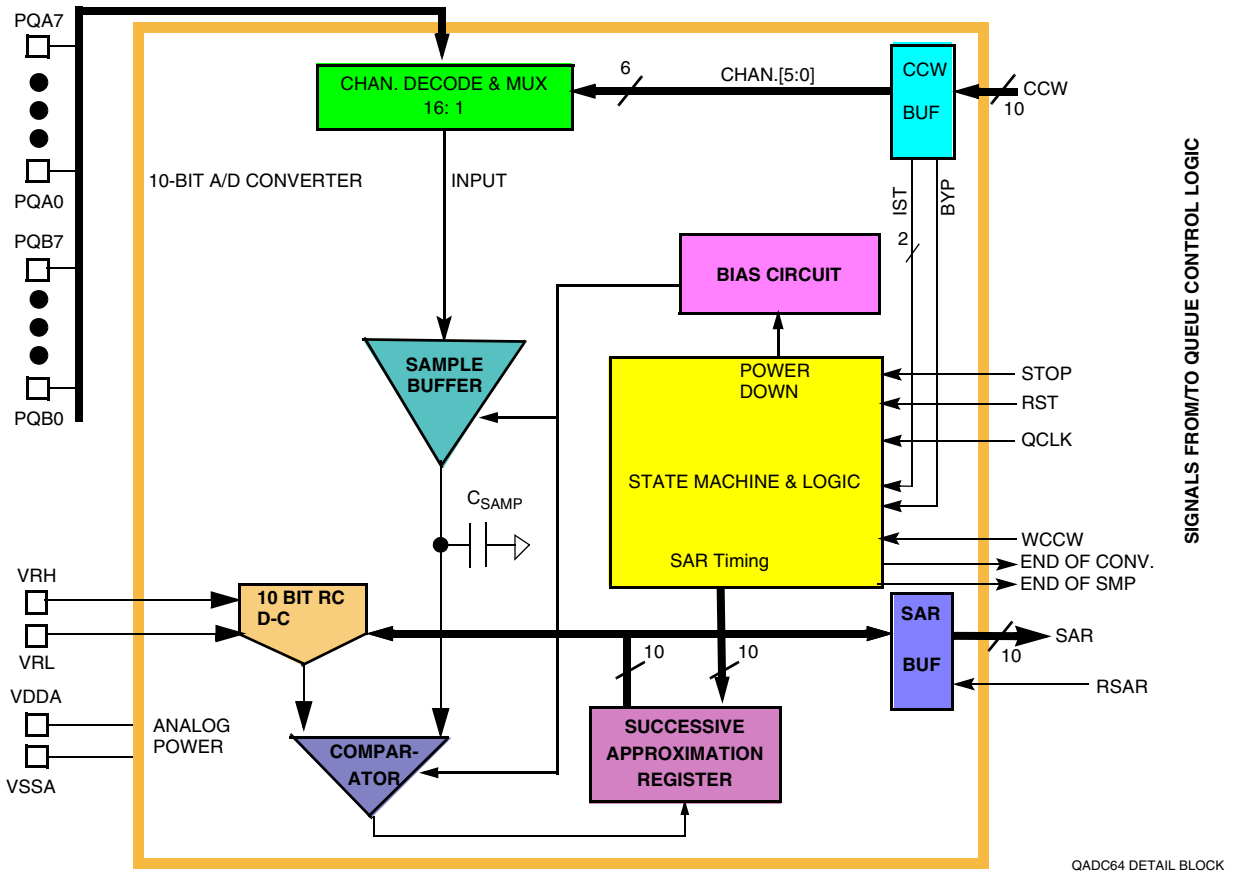


Figure 8-4 QADC64 Module Block Diagram

8.9.1 Conversion Cycle Times

Total conversion time is made up of initial sample time, final sample time, and resolution time. Initial sample time refers to the time during which the selected input channel is driven by the buffer amplifier onto the sample capacitor. The buffer amplifier can be disabled by means of the BYP bit in the CCW. During the final sampling period, amplifier is bypassed, and the multiplexer input charges the RC DAC array directly. During the resolution period, the voltage in the RC DAC array is converted to a digital value and stored in the SAR.

Initial sample time is fixed at two QCLK cycles. Final sample time can be 2, 4, 8, or 16 QCLK cycles, depending on the value of the IST field in the CCW. Resolution time is ten QCLK cycles.

Sample and resolution require a minimum of 14 QCLK clocks (7 μs with a 2 MHz QCLK). If the maximum final sample time period of 16 QCLKs is selected, the total conversion time is 13.0 μs with a 2 MHz QCLK.

Figure 8-5 illustrates the timing for conversions. This diagram assumes a final sampling period of two QCLK cycles.

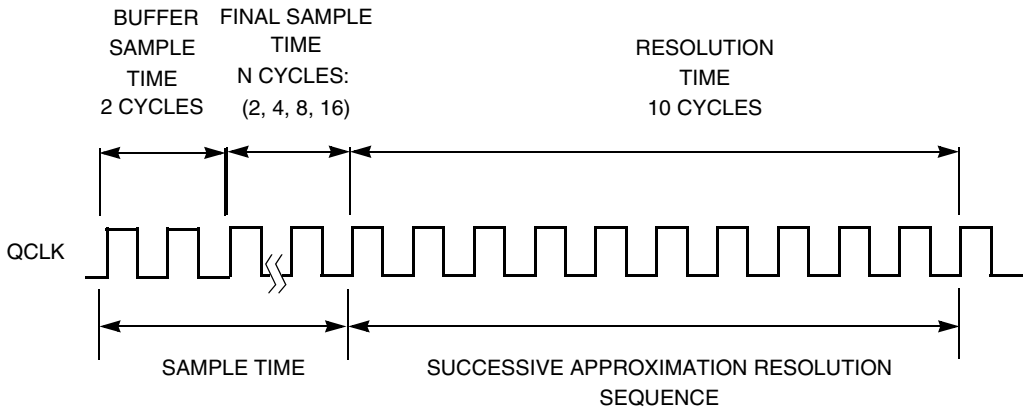


Figure 8-5 Conversion Timing

8.9.1.1 Amplifier Bypass Mode Conversion Timing

If the amplifier bypass mode is enabled for a conversion by setting the amplifier bypass (BYP) bit in the CCW, the timing changes to that shown in **Figure 8-6**. The buffered sample time is eliminated, reducing the potential conversion time by two QCLKs. However, due to internal RC effects, a minimum final sample time of four QCLKs must be allowed. This results in no savings of QCLKs. When using the bypass mode, the external circuit should be of low source impedance, typically less than 10 kΩ. Also, the loading effects of the external circuitry by the QADC64 need to be considered, since the benefits of the sample amplifier are not present.

NOTE

Because of internal RC time constants, a sample time of two QCKLs in bypass mode for high frequency operation is not recommended.

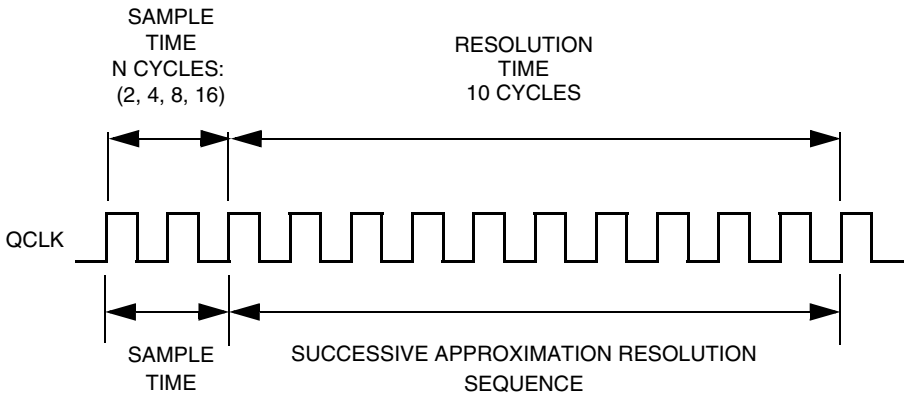


Figure 8-6 Bypass Mode Conversion Timing

8.9.2 Front-End Analog Multiplexer

The internal multiplexer selects one of the 16 analog input pins or one of three special internal reference channels for conversion. The following are the three special channels:

- V_{RH} — Reference voltage high
- V_{RL} — Reference voltage low
- $(V_{RH} - V_{RL})/2$ — Mid-reference voltage

The selected input is connected to one side of the DAC capacitor array. The other side of the DAC array is connected to the comparator input. The multiplexer also includes positive and negative stress protection circuitry, which prevents other channels from affecting the present conversion when excessive voltage levels are applied to the other channels. Refer to **APPENDIX E ELECTRICAL AND AC CHARACTERISTICS** for specific voltage level limits.

8.9.3 Digital-to-Analog Converter Array

The digital-to-analog converter (DAC) array consists of binary-weighted capacitors and a resistor-divider chain. The array serves two purposes:

- The array holds the sampled input voltage during conversion.
- The resistor-capacitor array provides the mechanism for the successive approximation A/D conversion.

Resolution begins with the MSB and works down to the LSB. The switching sequence is controlled by the SAR logic.

8.9.4 Comparator

The comparator is used during the approximation process to sense whether the digitally selected arrangement of the DAC array produces a voltage level higher or lower than the sampled input. The comparator output feeds into the SAR which accumulates the A/D conversion result sequentially, starting with the MSB.

8.9.5 Successive Approximation Register

The input of the successive approximation register (SAR) is connected to the comparator output. The SAR sequentially receives the conversion value one bit at a time, starting with the MSB. After accumulating the ten bits of the conversion result, the SAR data is transferred by the queue control logic in the digital section to the appropriate result location, where it may be read by user software.

8.10 Digital Control Subsystem

The digital control subsystem includes the clock and periodic/interval timer, control and status registers, the conversion command word table RAM, and the result word table RAM.

The central element for control of QADC64 conversions is the 64-entry conversion command word (CCW) table. Each CCW specifies the conversion of one input channel. Depending on the application, one or two queues can be established in the CCW

table. A queue is a scan sequence of one or more input channels. By using a pause mechanism, subqueues can be created within the two queues. Each queue can be operated using several different scan modes. The scan modes for queue 1 and queue 2 are programmed in QACR1 and QACR2. Once a queue has been started by a trigger event (any of the ways to cause the QADC64 to begin executing the CCWs in a queue or subqueue), the QADC64 performs a sequence of conversions and places the results in the result word table.

8.10.1 Queue Priority

Queue 1 has execution priority over queue 2 execution. **Table 8-3** shows the conditions under which queue 1 asserts its priority:

Table 8-3 Queue 1 Priority Assertion

Queue State	Result
Inactive	A trigger event for queue 1 or queue 2 causes the corresponding queue execution to begin.
Queue 1 active/trigger event occurs for queue 2	Queue 2 cannot begin execution until queue 1 reaches completion or the paused state. The status register records the trigger event by reporting the queue 2 status as trigger pending. Additional trigger events for queue 2, which occur before execution can begin, are recorded as trigger overruns.
Queue 2 active/trigger event occurs for queue 1	The current queue 2 conversion is aborted. The status register reports the queue 2 status as suspended. Any trigger events occurring for queue 2 while queue 2 is suspended are recorded as trigger overruns. Once queue 1 reaches the completion or the paused state, queue 2 begins executing again. The programming of the resume bit in QACR2 determines which CCW is executed in queue 2.
Simultaneous trigger events occur for queue 1 and queue 2	Queue 1 begins execution and the queue 2 status is changed to trigger pending.
Subqueues paused	The pause feature can be used to divide queue 1 and/or queue 2 into multiple subqueues. A subqueue is defined by setting the pause bit in the last CCW of the subqueue.

Figure 8-7 shows the CCW format and an example of using pause to create subqueues. Queue 1 is shown with four CCWs in each subqueue and queue 2 has two CCWs in each subqueue.

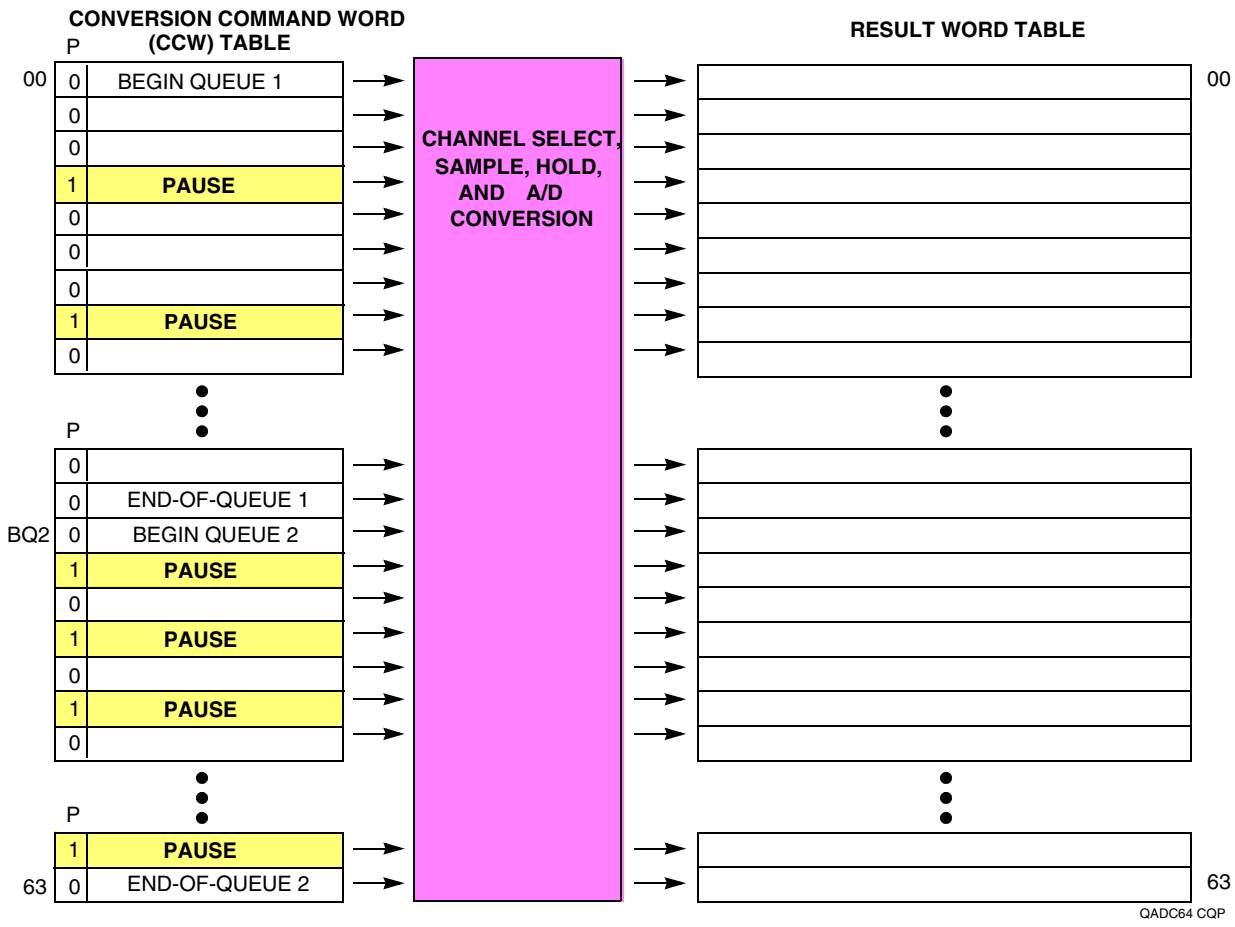


Figure 8-7 QADC64 Queue Operation with Pause

The queue operating mode selected for queue 1 determines what type of trigger event causes the execution of each of the subqueues within queue 1. Similarly, the queue operating mode for queue 2 determines the type of trigger event required to execute each of the subqueues within queue 2.

The choice of single-scan or continuous-scan applies to the full queue, and is not applied to each subqueue. Once a subqueue is initiated, each CCW is executed sequentially until the last CCW in the subqueue is executed and the pause state is entered. Execution can only continue with the next CCW, which is the beginning of the next subqueue. A subqueue cannot be executed a second time before the overall queue execution has been completed.

Trigger events which occur during the execution of a subqueue are ignored, except that the trigger overrun flag is set. When continuous-scan mode is selected, a trigger event occurring after the completion of the last subqueue (after the queue completion flag is set), causes execution to continue with the first subqueue, starting with the first CCW in the queue.

When the QADC64 encounters a CCW with the pause bit set, the queue enters the paused state after completing the conversion specified in the CCW with the pause bit. The pause flag is set and a pause software interrupt may optionally be issued. The status of the queue is shown to be paused, indicating completion of a subqueue. The QADC64 then waits for another trigger event to again begin execution of the next subqueue.

8.10.2 Queue Boundary Conditions

The following are queue operation boundary conditions:

- The first CCW in a queue contains channel 63, the end-of-queue (EOQ) code. The queue becomes active and the first CCW is read. The end-of-queue is recognized, the completion flag is set, and the queue becomes idle. A conversion is not performed.
- BQ2 (beginning of queue 2) is set at the end of the CCW table (63) and a trigger event occurs on queue 2. Refer to **8.12.5 QADC64 Control Register 0 (QACR0)** for information on BQ2. The end-of-queue condition is recognized, a conversion is performed, the completion flag is set, and the queue becomes idle.
- BQ2 is set to CCW0 and a trigger event occurs on queue 1. After reading CCW0, the end-of-queue condition is recognized, the completion flag is set, and the queue becomes idle. A conversion is not performed.
- BQ2 (beginning of queue 2) is set beyond the end of the CCW table (64–127) and a trigger event occurs on queue 2. Refer to **8.12.7 QADC64 Control Register 2 (QACR2)** for information on BQ2. The end-of-queue condition is recognized immediately, the completion flag is set, and the queue becomes idle. A conversion is not performed.

NOTE

Multiple end-of-queue conditions may be recognized simultaneously, although there is no change in the QADC64 behavior. For example, if BQ2 is set to CCW0, CCW0 contains the EOQ code, and a trigger event occurs on queue 1, the QADC64 reads CCW0 and detects both end-of-queue conditions. The completion flag is set for queue 1 only and it becomes idle.

Boundary conditions also exist for combinations of pause and end-of-queue. One case is when a pause bit is in one CCW and an end-of-queue condition is in the next CCW. The conversion specified by the CCW with the pause bit set completes normally. The pause flag is set. However, since the end-of-queue condition is recognized, the completion flag is also set and the queue status becomes idle, not paused. Examples of this situation include:

- The pause bit is set in CCW5 and the channel 63 (EOQ) code is in CCW6.
- The pause bit is set in CCW63.
- During queue 1 operation, the pause bit is set in CCW14 and BQ2 points to CCW15.

Another pause and end-of-queue boundary condition occurs when the pause and an end-of-queue condition occur in the same CCW. Both the pause and end-of-queue conditions are recognized simultaneously. The end-of-queue condition has precedence so a conversion is not performed for the CCW and the pause flag is not set. The QADC64 sets the completion flag and the queue status becomes idle. Examples of this situation are:

- The pause bit is set in CCW0 and EOQ is programmed into CCW0.
- During queue 1 operation, the pause bit is set in CCW20, which is also BQ2.

8.10.3 Scan Modes

The QADC64 queuing mechanism provides several methods for automatically scanning input channels. In single-scan mode, a single pass through a sequence of conversions defined by a queue is performed. In continuous-scan mode, multiple passes through a sequence of conversions defined by a queue are executed. The possible modes are:

- Disabled and reserved mode
- Software initiated single-scan mode
- External trigger single-scan mode
- External gated single-scan mode (queue 1 only)
- Interval timer single-scan mode
- Software initiated continuous-scan mode
- External trigger continuous-scan mode
- External gated continuous-scan mode (queue 1 only)
- Interval timer continuous-scan mode

The following paragraphs describe the disabled/reserved, single-scan, and continuous-scan operations.

8.10.3.1 Disabled Mode

When the disabled mode is selected, the queue is not active. Trigger events cannot initiate queue execution. When both queue 1 and queue 2 are disabled, wait states are not encountered for IMB accesses of the RAM. When both queues are disabled, it is safe to change the QCLK prescaler values.

8.10.3.2 Reserved Mode

Reserved mode allows for future mode definitions. When the reserved mode is selected, the queue is not active.

CAUTION

Do not use a reserved mode. Unspecified operations may result.

8.10.3.3 Single-Scan Modes

When the application software wants to execute a single pass through a sequence of conversions defined by a queue, a single-scan queue operating mode is selected. By programming the MQ field in QACR1 or QACR2, the following modes can be selected:

- Software initiated single-scan mode
- External trigger single-scan mode
- External gated single-scan mode (queue 1 only)
- Interval timer single-scan mode

NOTE

Queue 2 can not be programmed for external gated single-scan mode.

In all single-scan queue operating modes, the software must also enable the queue to begin execution by writing the single-scan enable bit to a one in the queue's control register. The single-scan enable bits, SSE1 and SSE2, are provided for queue 1 and queue 2 respectively.

Until the single-scan enable bit is set, any trigger events for that queue are ignored. The single-scan enable bit may be set to a one during the write cycle, which selects the single-scan queue operating mode. The single-scan enable bit can be written as a one or a zero, but is always read as a zero. The completion flag, completion interrupt, or queue status are used to determine when the queue has completed.

After the single-scan enable bit is set, a trigger event causes the QADC64 to begin execution with the first CCW in the queue. The single-scan enable bit remains set until the queue is completed. After the queue reaches completion, the QADC64 resets the single-scan enable bit to zero. If the single-scan enable bit is written to a one or a zero by the software before the queue scan is complete, the queue is not affected. However, if the software changes the queue operating mode, the new queue operating mode and the value of the single-scan enable bit are recognized immediately. The conversion in progress is aborted and the new queue operating mode takes effect.

In the software initiated single-scan mode, the writing of a 1 to the single-scan enable bit causes the QADC64 to internally generate a trigger event and the queue execution begins immediately. In the other single-scan queue operating modes, once the single-scan enable bit is written, the selected trigger event must occur before the queue can start. The single-scan enable bit allows the entire queue to be scanned once. A trigger overrun is captured if a trigger event occurs during queue execution in the external trigger single-scan mode and the interval timer single-scan mode.

In the interval timer single-scan mode, the next expiration of the timer is the trigger event for the queue. After the queue execution is complete, the queue status is shown as idle. The software can restart the queue by setting the single-scan enable bit to a 1. Queue execution begins with the first CCW in the queue.

Software Initiated Single-Scan Mode. Software can initiate the execution of a scan sequence for queue 1 or 2 by selecting the software initiated single-scan mode, and writing the single-scan enable bit in QACR1 or QACR2. A trigger event is generated internally and the QADC64 immediately begins execution of the first CCW in the queue. If a pause occurs, another trigger event is generated internally, and then execution continues without pausing.

The QADC64 automatically performs the conversions in the queue until an end-of-queue condition is encountered. The queue remains idle until the software again sets the single-scan enable bit. While the time to internally generate and act on a trigger event is very short, software can momentarily read the status conditions, indicating that the queue is paused. The trigger overrun flag is never set while in the software initiated single-scan mode.

The software initiated single-scan mode is useful in the following applications:

- Allows software complete control of the queue execution.
- Allows the software to easily alternate between several queue sequences.

External Trigger Single-Scan Mode. The external trigger single-scan mode is a variation of the external trigger continuous-scan mode, and is also available with both queue 1 and queue 2. The software programs the polarity of the external trigger edge that is to be detected, either a rising or a falling edge. The software must enable the scan to occur by setting the single-scan enable bit for the queue.

The first external trigger edge causes the queue to be executed one time. Each CCW is read and the indicated conversions are performed until an end-of-queue condition is encountered. After the queue is completed, the QADC64 clears the single-scan enable bit. Software may set the single-scan enable bit again to allow another scan of the queue to be initiated by the next external trigger edge.

The external trigger single-scan mode is useful when the input trigger rate can exceed the queue execution rate. Analog samples can be taken in sync with an external event, even though the software is not interested in data taken from every edge. The software can start the external trigger single-scan mode and get one set of data, and at a later time, start the queue again for the next set of samples.

When a pause bit is encountered during external trigger single-scan mode, another trigger event is required for queue execution to continue. Software involvement is not needed to enable queue execution to continue from the paused state.

The external trigger single-scan mode is also useful when the software needs to change the polarity of the external trigger so that both the rising and falling edges cause queue execution.

External Gated Single-Scan Mode. The QADC64 provides external gating for queue 1 only. When external gated single-scan mode is selected, a transition on the associated external trigger pin initiates queue execution. The polarity of the external gated signal is fixed so only a high level opens the gate and a low level closes the gate. Once the gate is open, each CCW is read and the indicated conversions are performed until the gate is closed. Software must enable the scan to occur by setting the single-scan enable bit for queue 1. If a pause in a CCW is encountered, the pause flag **will not set**, and execution continues without pausing.

While the gate is open, queue 1 executes one time. Each CCW is read and the indicated conversions are performed until an end-of-queue condition is encountered. When queue 1 completes, the QADC64 sets the completion flag (CF1) and clears the

single-scan enable bit. Software may set the single-scan enable bit again to allow another scan of queue 1 to be initiated during the next open gate.

If the gate closes before queue 1 completes execution, the current CCW completes, execution of queue 1 stops, the single-scan enable bit is cleared, and the PF1 bit is set. Software can read the CWPQ1 to determine the last valid conversion in the queue. Software must set the single-scan enable bit again and should clear the PF1 bit before another scan of queue 1 is initiated during the next open gate. The start of queue 1 is always the first CCW in the CCW table.

Interval Timer Single-Scan Mode. Both queues can use the periodic/interval timer in a single-scan queue operating mode. The timer interval can range from 128 to 128K QCLK cycles in binary multiples. When the interval timer single-scan mode is selected and the software sets the single-scan enable bit in QACR1(2), the timer begins counting. When the time interval elapses, an internal trigger event is created to start the queue and the QADC64 begins execution with the first CCW.

The QADC64 automatically performs the conversions in the queue until a pause or an end-of-queue condition is encountered. When a pause occurs, queue execution stops until the timer interval elapses again, and queue execution continues. When the queue execution reaches an end-of-queue situation the single-scan enable bit is cleared. Software may set the single-scan enable bit again, allowing another scan of the queue to be initiated by the interval timer.

The interval timer generates a trigger event whenever the time interval elapses. The trigger event may cause the queue execution to continue following a pause, or may be considered a trigger overrun. Once the queue execution is completed, the single-scan enable bit must be set again to enable the timer to count again.

Normally, only one queue will be enabled for interval timer single-scan mode and the timer will reset at the end-of-queue. However, if both queues are enabled for either single-scan or continuous interval timer mode, the end-of-queue condition will not reset the timer while the other queue is active. In this case, the timer will reset when both queues have reached end-of-queue.

The interval timer single-scan mode can be used in applications which need coherent results, for example:

- When it is necessary that all samples are guaranteed to be taken during the same scan of the analog pins.
- When the interrupt rate in the periodic timer continuous-scan mode would be too high.
- In sensitive battery applications, where the single-scan mode uses less power than the software initiated continuous-scan mode.

8.10.3.4 Continuous-Scan Modes

When the application software wants to execute multiple passes through a sequence of conversions defined by a queue, a continuous-scan queue operating mode is

selected. By programming the MQ1(2) field in QACR1(2), the following software initiated modes can be selected:

- Software initiated continuous-scan mode
- External trigger continuous-scan mode
- External gated continuous-scan mode (queue 1 only)
- Interval timer continuous-scan mode

When a queue is programmed for a continuous-scan mode, the single-scan enable bit in the queue control register does not have any meaning or effect. As soon as the queue operating mode is programmed, the selected trigger event can initiate queue execution.

In the case of the software initiated continuous-scan mode, the trigger event is generated internally and queue execution begins immediately. In the other continuous-scan queue operating modes, the selected trigger event must occur before the queue can start. A trigger overrun is captured if a trigger event occurs during queue execution in the external trigger continuous-scan mode and the periodic timer continuous-scan mode.

After the queue execution is complete, the queue status is shown as idle. Since the continuous-scan queue operating modes allow the entire queue to be scanned multiple times, software involvement is not needed to enable queue execution to continue from the idle state. The next trigger event causes queue execution to begin again, starting with the first CCW in the queue.

NOTE

In this version of QADC64, coherent samples can be guaranteed. The time between consecutive conversions has been designed to be consistent, provided the sample time bits in both the CCW and IST are identical. However, there is one exception. For queues that end with a CCW containing EOQ code (channel 63), the last queue conversion to the first queue conversion requires 1 additional CCW fetch cycle. Therefore continuous samples are not coherent at this boundary.

In addition, the time from trigger to first conversion can not be guaranteed since it is a function of clock synchronization, programmable trigger events, queue priorities, and other factors.

Software Initiated Continuous-Scan Mode. When the software initiated continuous-scan mode is programmed, the trigger event is generated automatically by the QADC64. Queue execution begins immediately. If a pause is encountered, another trigger event is generated internally, and then execution continues without pausing. When the end-of-queue is reached, another internal trigger event is generated, and queue execution begins again from the beginning of the queue.

While the time to internally generate and act on a trigger event is very short, software can momentarily read the status conditions, indicating that the queue is idle. The trigger overrun flag is never set while in the software initiated continuous-scan mode.

The software initiated continuous-scan mode keeps the result registers updated more frequently than any of the other queue operating modes. The software can always read the result table to get the latest converted value for each channel. The channels scanned are kept up to date by the QADC64 without software involvement. Software can read a result value at any time.

The software initiated continuous-scan mode may be chosen for either queue, but is normally used only with queue 2. When the software initiated continuous-scan mode is chosen for queue 1, that queue operates continuously and queue 2, being lower in priority, never gets executed. The short interval of time between a queue 1 completion and the subsequent trigger event is not sufficient to allow queue 2 execution to begin.

The software initiated continuous-scan mode is a useful choice with queue 2 for converting channels that do not need to be synchronized to anything, or for the slow-to-change analog channels. Interrupts are normally not used with the software initiated continuous-scan mode. Rather, the software reads the latest conversion result from the result table at any time. Once initiated, software action is not needed to sustain conversions of channel. Data read at different locations, however, may or may not be coherent (that is, from the same queue scan sequence).

External Trigger Continuous-Scan Mode. The QADC64 provides external trigger pins for both queues. When the external trigger software initiated continuous-scan mode is selected, a transition on the associated external trigger pin initiates queue execution. The polarity of the external trigger signal is programmable, so that the software can choose to begin queue execution on the rising or falling edge. Each CCW is read and the indicated conversions are performed until an end-of-queue condition is encountered. When the next external trigger edge is detected, the queue execution begins again automatically. Software initialization is not needed between trigger events.

When a pause bit is encountered in external trigger continuous-scan mode, another trigger event is required for queue execution to continue. Software involvement is not needed to enable queue execution to continue from the paused state.

Some applications need to synchronize the sampling of analog channels to external events. There are cases when it is not possible to use software initiation of the queue scan sequence, since interrupt response times vary.

External Gated Continuous-Scan Mode. The QADC64 provides external gating for queue 1 only. When external gated continuous-scan mode is selected, a transition on the associated external trigger pin initiates queue execution. The polarity of the external gated signal is fixed so a high level opens the gate and a low level closes the gate. Once the gate is open, each CCW is read and the indicated conversions are performed until the gate is closed. When the gate opens again, the queue execution automatically begins again from the beginning of the queue. Software initialization is not needed between trigger events. If a pause in a CCW is encountered, the pause flag **will not set**, and execution continues without pausing.

The purpose of external gated continuous-scan mode is to continuously collect digitized samples while the gate is open and to have the most recent samples available. To ensure consistent sample times in waveform digitizing, for example, the programmer must ensure that all CCW's have identical sample time settings in IST.

It is up to the programmer to ensure that the queue is large enough so that a maximum gate open time will not reach an end-of-queue. However it is useful to take advantage of a smaller queue in the manner described below.

In the event that the queue completes before the gate closes, a completion flag will be set and the queue will roll over to the beginning and continue conversions until the gate closes. If the gate remains open and the queue completes a second time, the trigger overrun flag will be set and the queue will roll over again. The queue will continue to execute until the gate closes or the mode is disabled.

If the gate closes before queue 1 completes execution, the current CCW completes and execution of queue 1 stops and QADC64 sets the PF1 bit to indicate an incomplete queue. Software can read the CWPQ1 to determine the last valid conversion in the queue. In this mode, if the gate opens again, execution of queue 1 begins again. The start of queue 1 is always the first CCW in the CCW table.

Interval Timer Continuous-Scan Mode. The QADC64 includes a dedicated periodic/interval timer for initiating a scan sequence on queue 1 and/or queue 2. Software selects a programmable timer interval ranging from 128 to 128K times the QCLK period in binary multiples. The QCLK period is prescaled down from the intermodule bus (IMB) MCU clock.

When a periodic timer continuous-scan mode is selected for queue 1 and/or queue 2, the timer begins counting. After the programmed interval elapses, the timer generated trigger event starts the appropriate queue. Meanwhile, the QADC64 automatically performs the conversions in the queue until an end-of-queue condition or a pause is encountered. When a pause occurs, the QADC64 waits for the periodic interval to expire again, then continues with the queue. Once end-of-queue has been detected, the next trigger event causes queue execution to begin again with the first CCW in the queue.

The periodic timer generates a trigger event whenever the time interval elapses. The trigger event may cause the queue execution to continue following a pause or queue completion, or may be considered a trigger overrun. As with all continuous-scan queue operating modes, software action is not needed between trigger events.

Software enables the completion interrupt when using the periodic timer continuous-scan mode. When the interrupt occurs, the software knows that the periodically collected analog results have just been taken. The software can use the periodic interrupt to obtain non-analog inputs as well, such as contact closures, as part of a periodic look at all inputs.

8.10.4 QADC64 Clock (QCLK) Generation

Figure 8-8 is a block diagram of the clock subsystem. The QCLK provides the timing for the A/D converter state machine, which controls the timing of the conversion. The QCLK is also the input to a 17-stage binary divider which implements the periodic/interval timer. To retain the specified analog conversion accuracy, the QCLK frequency (F_{QCLK}) must be within the tolerance specified in **APPENDIX E ELECTRICAL AND AC CHARACTERISTICS**.

Before using the QADC64, the software must initialize the prescaler with values that put the QCLK within the specified range. Though most software applications initialize the prescaler once and do not change it, write operations to the prescaler fields are permitted.

NOTE

For software compatibility with earlier versions of QADC64, the definition of PSL, PSH, and PSA have been maintained. However, the requirements on minimum time and minimum low time no longer exist.

CAUTION

A change in the prescaler value while a conversion is in progress is likely to corrupt the result from any conversion in progress. Therefore, any prescaler write operation should be done only when both queues are in the disabled modes.

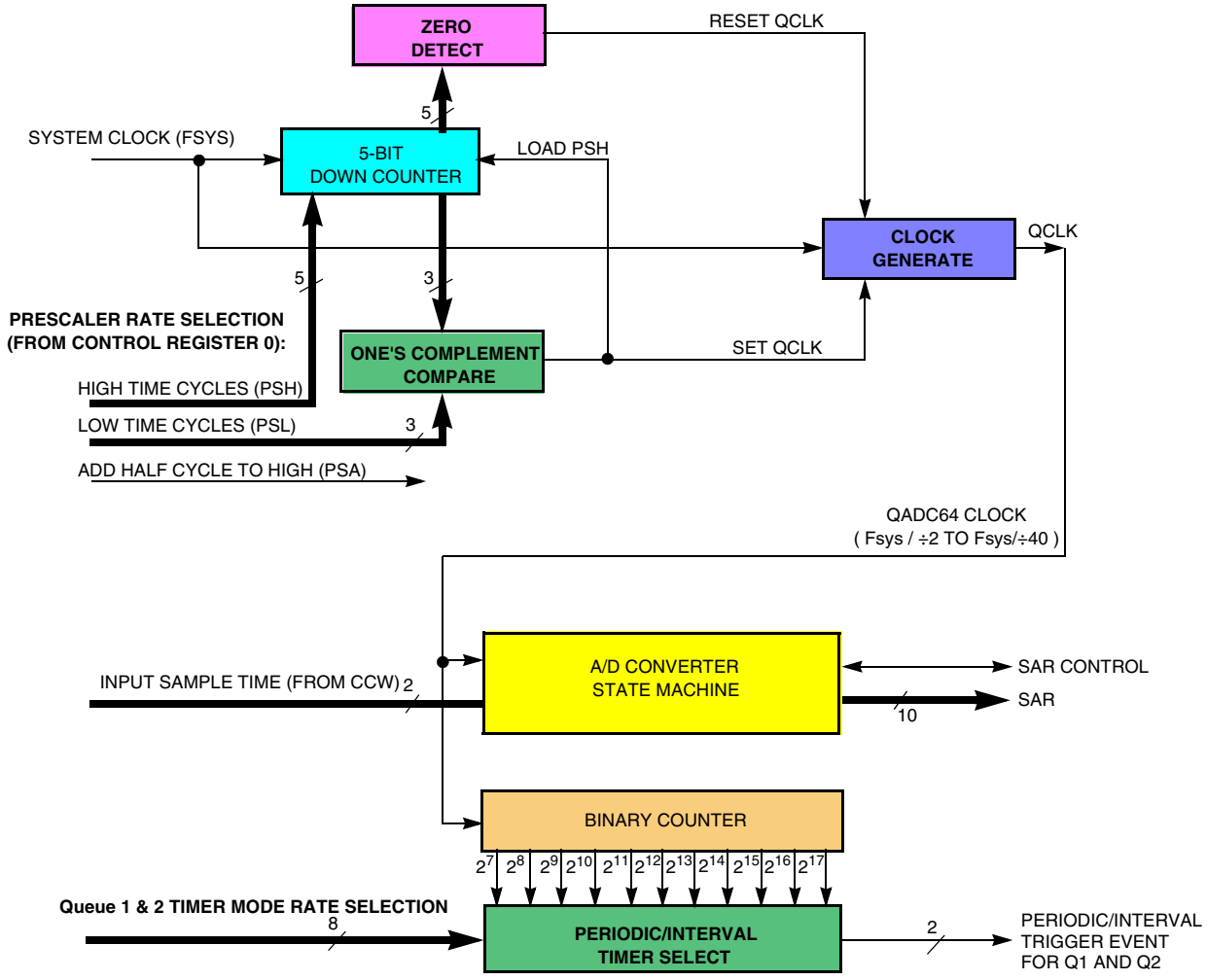


Figure 8-8 QADC64 Clock Subsystem Functions

To accommodate wide variations of the main MCU clock frequency (IMB system clock – F_{SYS}), QCLK is generated by a programmable prescaler which divides the MCU system clock to a frequency within the specified QCLK tolerance range. To allow the A/D conversion time to be maximized across the spectrum of system clock frequencies, the QADC64 prescaler permits the frequency of QCLK to be software selectable. It also allows the duty cycle of the QCLK waveform to be programmable.

The software establishes the basic high phase of the QCLK waveform with the PSH (prescaler clock high time) field in QACR0, and selects the basic low phase of QCLK with the PSL (prescaler clock low time) field. The combination of the PSH and PSL parameters establishes the frequency of the QCLK.

NOTE

The guideline for selecting PSH and PSL is select is to maintain approximately 50% duty cycle. So for prescaler values less then 16, or PSH ≈ PSL. For prescaler values greater than 16 keep PSL as large as possible.

Figure 8-8 shows that the prescaler is essentially a variable pulse width signal generator. A 5-bit down counter, clocked at the system clock rate, is used to create both the high phase and the low phase of the QCLK signal. At the beginning of the high phase, the 5-bit counter is loaded with the 5-bit PSH value. When the zero detector finds that the high phase is finished, the QCLK is reset. A 3-bit comparator looks for a one's complement match with the 3-bit PSL value, which is the end of the low phase of the QCLK. The PSA bit was maintained for software compatibility, but has no effect on QADC64.

The following equations define Qclk frequency:

$$\text{High QCLK Time} = (\text{PSH} + 1) \div F_{SYS}$$

$$\text{Low QCLK Time} = (\text{PSL} + 1) \div F_{SYS}$$

$$F_{QCLK} = 1 \div (\text{High QCLK Time} + \text{Low QCLK Time})$$

Where:

- PSH = 0 to 31, the prescaler QCLK high cycles in QACR0
- PSL = 0 to 7, the prescaler QCLK low cycles in QACR0
- F_{SYS} = System clock frequency
- F_{QCLK} = QCLK frequency

The following are equations for calculating the QCLK high and low phases in example 1 shown in **Figure 8-9**:

$$\text{High QCLK Time} = (11 + 1) \div 40 \times 10^6 = 300 \text{ ns}$$

$$\text{Low QCLK Time} = (7 + 1) \div 40 \times 10^6 = 200 \text{ ns}$$

$$F_{QCLK} = 1/(300 + 200) = 2 \text{ Mhz}$$

The following are equations for calculating the QCLK high and low phases in example 2 shown in **Figure 8-9**:

$$\text{High QCLK Time} = (7 + 1) \div 32 \times 10^6 = 250 \text{ ns}$$

$$\text{Low QCLK Time} = (7 + 1) \div 32 \times 10^6 = 250 \text{ ns}$$

$$F_{QCLK} = 1/(250 + 250) = 2 \text{ Mhz}$$

Figure 8-9 and **Table 8-4** show examples of QCLK programmability. The examples include conversion times based on the following assumption:

- Input sample time is as fast as possible (IST = 0, 2 QCLK cycles).

Figure 8-9 and Table 8-4 also show the conversion time calculated for a single conversion in a queue. For other MCU system clock frequencies and other input sample times, the same calculations can be made.

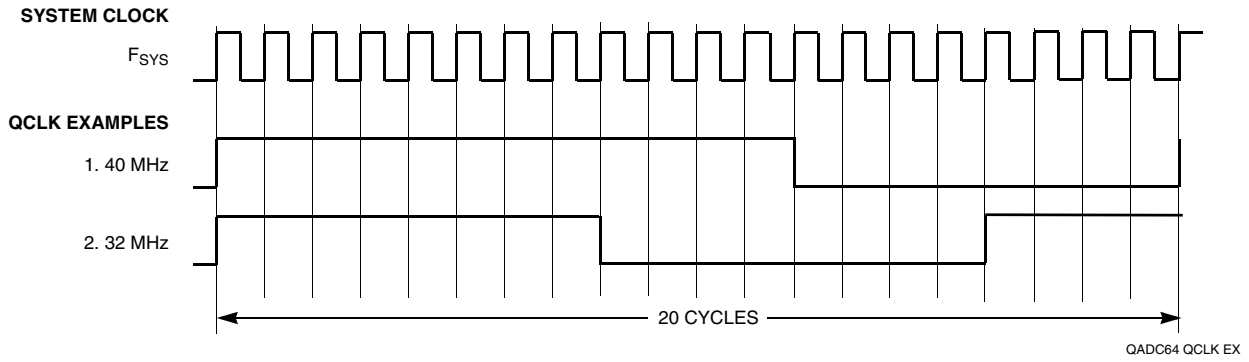


Figure 8-9 QADC64 Clock Programmability Examples

Table 8-4 QADC64 Clock Programmability

Control Register 0 Information					Input Sample Time (IST) = 0b00	
Example Number	Frequency	PSH	PSA	PSL	QCLK (MHz)	Conversion Time (µs)
1	40 Mhz	11	0	7	2.0	7.0
2	32 Mhz	7	0	7	2.0	7.0

NOTE

PSA is maintained for software compatibility but has no functional benefit to this version of the module.

The MCU system clock frequency is the basis of the QADC64 timing. The QADC64 requires that the system clock frequency be at least twice the QCLK frequency. The QCLK frequency is established by the combination of the PSH and PSL parameters in QACR0. The 5-bit PSH field selects the number of system clock cycles in the high phase of the QCLK wave. The 3-bit PSL field selects the number of system clock cycles in the low phase of the QCLK wave.

Example 1 in Figure 8-9 shows that when PSH = 11, the QCLK remains high for twelve cycles of the system clock. It also shows that when PSL = 7, the QCLK remains low for eight system clock cycles. In example 2, PSH = 7, the QCLK remains high for eight cycles of the system clock. It also shows that when PSL = 7, the QCLK remains low for eight system clock cycles.

8.10.5 Periodic/Interval Timer

The on-chip periodic/interval timer is enabled to generate trigger events at a programmable interval, initiating execution of queue 1 and/or 2. The periodic/interval timer stays reset under the following conditions:

- Queue 1 and queue 2 are programmed to any queue operating mode which does not use the periodic/interval timer
- Interval timer single-scan mode is selected, but the single-scan enable bit is set to zero
- IMB system reset or the master reset is asserted
- Stop mode is selected
- Freeze mode is selected

Two other conditions which cause a pulsed reset of the timer are:

- Roll over of the timer counter
- A queue operating mode change from one periodic/interval timer mode to another periodic/interval timer mode, depending on which queues are active in timer mode.

NOTE

The periodic/interval timer will not reset for a queue 2 operating mode change from one periodic/interval timer mode to another periodic/interval timer mode while queue 1 is in an active periodic/interval timer mode.

During the low power stop mode, the periodic/interval timer is held in reset. Since low power stop mode causes QACR1 and QACR2 to be reset to zero, a valid periodic or interval timer mode must be written after stop mode is exited to release the timer from reset.

When the IMB internal FREEZE line is asserted and a periodic or interval timer mode is selected, the timer counter is reset after the conversion in progress completes. When the periodic or interval timer mode has been enabled (the timer is counting), but a trigger event has not been issued, the freeze mode takes effect immediately, and the timer is held in reset. When the internal FREEZE line is negated, the timer counter starts counting from the beginning.

8.11 Interrupts

Interrupt recognition and servicing involve interaction between the integration module, the CPU, and the module requesting interrupt service. This section provides an overview of the QADC interrupt process. Polled operation, an alternative to using interrupts, is discussed along with the different aspects of interrupt operation.

An interrupt is a special form of exception processing. Interrupt requests can be generated on-chip, or can come from external sources. However, the CPU services all interrupt requests as though originated by an on-chip module; to the CPU, an external interrupt request appears to come from the integration module. There are schemes to prioritize all interrupt requests and to arbitrate between simultaneous requests of the same priority. The QADC is configured to support interrupt acknowledge (IACK) cycles and vector generation.

8.11.1 Interrupt Operation

Figure 8-10 displays the QADC64 interrupt flow.

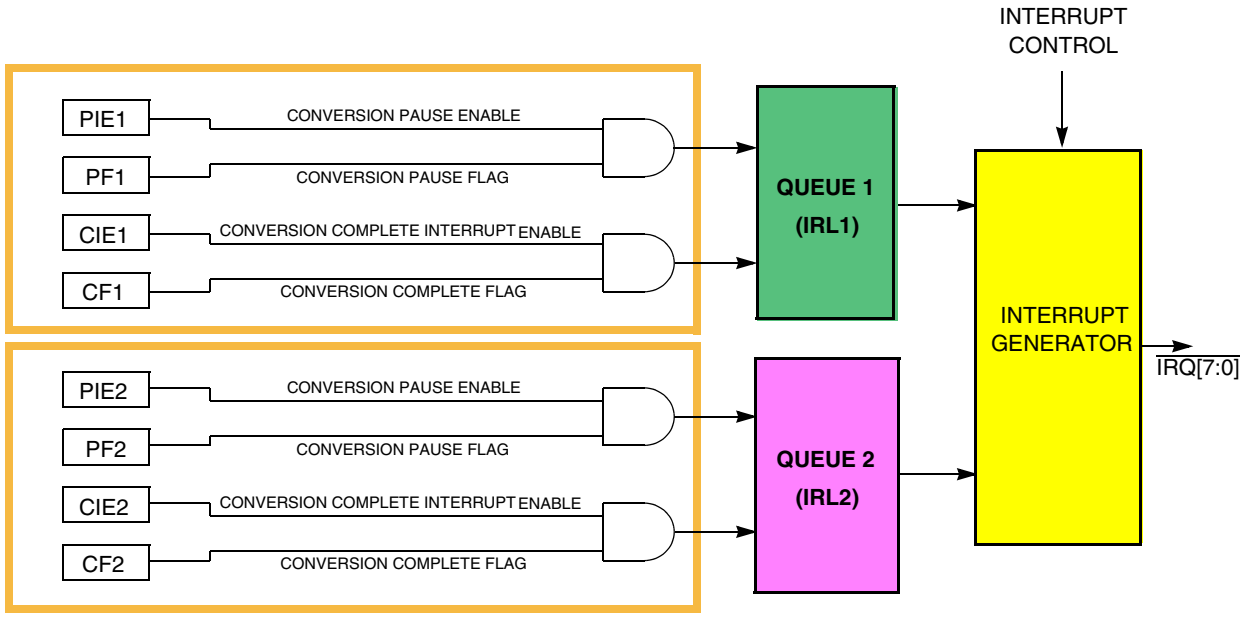


Figure 8-10 QADC64 Interrupt Flow Diagram

8.11.1.1 Polled and Interrupt-Driven Operation

QADC inputs can be monitored by polling or by using interrupts. When interrupts are not needed, software can disable the pause and completion interrupts and monitor the completion flag and the pause flag for each queue in the status register (QASR). In other words, flag bits can be polled to determine when new results are available.

Table 8-5 displays the status flag and interrupt enable bits which correspond to queue 1 and queue 2 activity. If interrupts are enabled for an event, the QADC requests interrupt service when the event occurs. Using interrupts does not require continuously polling the status flags to see if an event has taken place. However, status flags must be cleared after an interrupt is serviced, in order to disable the interrupt request. In both polled and interrupt-driven operating modes, status flags must be re-enabled after an event occurs. Flags are re-enabled by clearing appropriate QASR bits in a particular sequence. The register must first be read, then zeros must be written to the flags that are to be cleared. If a new event occurs betwe en the time that the register is read and the time that it is written, the associate d flag is not cleared.

8.11.2 Interrupt Sources

The QADC includes four sources of interrupt service requests, each of which is separately enabled. Each time the result is written for the last conversion command word (CCW) in a queue, the completion flag for the corresponding queue is set, and when enabled, an interrupt request is generated. In the same way, each time the result is

written for a CCW with the pause bit set, the queue pause flag is set, and when enabled, an interrupt request is generated.

Table 8-5 displays the status flag and interrupt enable bits which correspond to queue 1 and queue 2 activity. The pause and complete interrupts for queue 1 and queue 2 have separate interrupt vector levels, so that each source can be separately serviced.

Table 8-5 QADC64 Status Flags and Interrupt Sources

Queue	Queue Activity	Status Flag	Interrupt Enable Bit
Queue 1	Result written for the last CCW in queue 1	CF1	CIE1
	Result written for a CCW with pause bit set in queue 1	PF1	PIE1
Queue 2	Result written for the last CCW in queue 2	CF2	CIE2
	Result written for a CCW with pause bit set in queue 2	PF2	PIE2

8.11.3 Interrupt Priority

Interrupt priority is determined with a three-bit interrupt priority mask that is located in the bus master condition code register or status register. The interrupt priority mask can have eight possible values, from 0b000 to 0b111.

There are seven levels of interrupt priority, one to seven, each corresponding to a particular interrupt request signal. The bus master compares the priority of each interrupt service request to the mask value. Interrupt request levels greater than the mask value are accepted; interrupt request levels less than or equal to the mask value are ignored, except for the nonmaskable level seven interrupt request, which is serviced even if the bus master interrupt mask value is seven.

The values contained in the IRL1 and IRL2 fields in the interrupt register (QADC64INT) determine the priority of QADC interrupt service requests. A value of 0b000 in either field disables the interrupts associated with that field. IRL1 determines the priority of both queue 1 interrupt sources. IRL2 determines the priority of both queue 2 interrupt sources. As a result, queue 1 and queue 2 can have different priorities in the overall interrupt hierarchy of the MCU. The QADC also has an internal interrupt request prioritization. Queue1 interrupt requests are higher in priority than queue 2 requests, and completion flag requests are higher in priority than pause requests.

8.11.4 Interrupt Arbitration

After queue 1 or queue 2 issues an interrupt service request, the bus master performs an interrupt acknowledge cycle. During the interrupt acknowledge cycle, the bus master identifies the interrupt request level being acknowledged by placing it on the address bus. The QADC compares the acknowledged interrupt level with IRL1 and IRL2 values and responds if the values match.

The same interrupt priority level can be assigned to more than one module. For example, the QADC and the queued serial module (QSM) can both be assigned priority five. If the QADC and the QSM request interrupt service simultaneously, then the interrupt

arbitration (IARB) fields in the respective module configuration registers are used to determine which module is serviced first.

The IARB field is essentially a second-level priority in case of a tie. Each module that can request interrupt service has an IARB field. Arbitration is per formed by means of serial contention of IARB field bit values. Arbitration always takes place, even when a single source requests service.

IARB fields contain four bits. An IARB value of 0b1111 has the highest arbitration priority, and 0b0001 has the lowest. If a module with an IARB field value of 0b0000 requests interrupt service, the bus master processes a spurious interrupt exception because the module requesting the interrupt service cannot confirm that it made the request. Initialization software must assign each IARB field a unique non-zero value in order to implement the arbitration scheme. If two or more modules are assigned the same non-zero IARB field value, operation is undefined when interrupts of the same priority level are recognized.

8.11.5 Interrupt Vectors

When the QADC is the only module with an interrupt request pending at the level being acknowledged, or when the QADC IARB value is higher than that of other modules with requests pending at the acknowledged level, the QADC responds to the interrupt acknowledge cycle with an 8-bit interrupt vector number. The CPU uses the vector number to calculate a displacement into the exception vector table, then uses the vector at that location to jump to an interrupt service routine.

The interrupt vector base (IVB) field establishes the six high-order bits of the 8-bit interrupt vector number, and the QADC provides two low-order bits which correspond to one of the four internal QADC interrupt sources. **Figure 8-11** shows the format of the interrupt vector, and lists the binary coding of the two low-order bits for the four QADC interrupt sources.

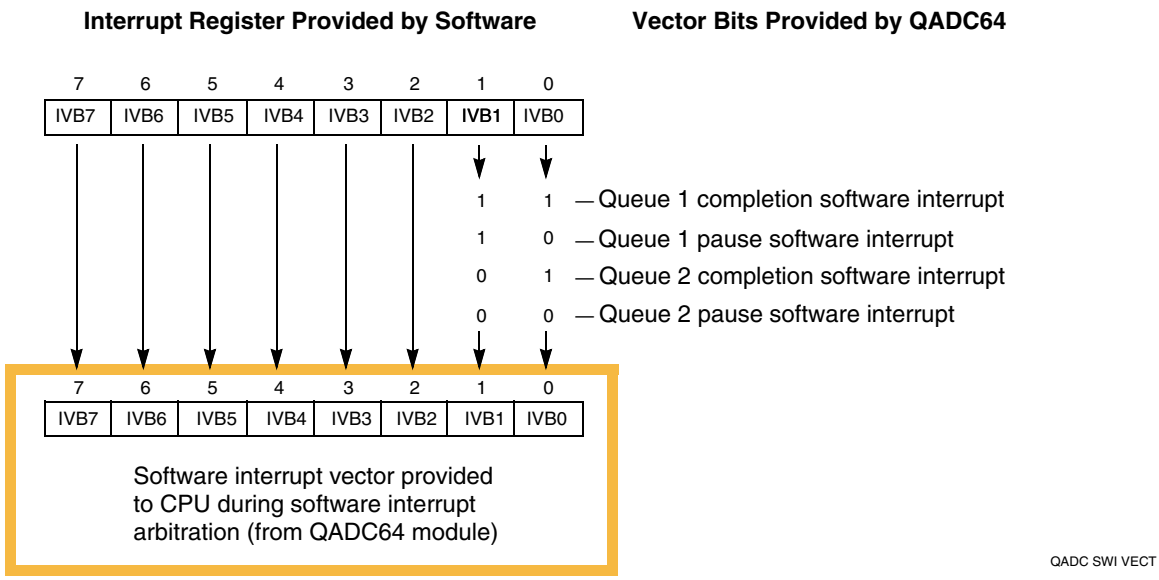


Figure 8-11 QADC64 Interrupt Vector Format

8.12 Programming Model

Each QADC64 occupies 1 Kbyte (512 16-bit entries) of address space. The address space consists of ten 16-bit control, status, and port registers; 64 16-bit entries in the CCW table; and 64 16-bit entries in the result table. The result table occupies 192 16-bit address locations because the result data is readable in three data alignment formats.

Table 8-6 shows the QADC64 memory map. The lowercase “x” appended to each register name represents “A” or “B” for the QADC64_A or QADC64_B module, respectively. The address is the base address of the module.

Table 8-6 QADC64 Address Map

Access	Address	MSB 15	LSB 0
S[1]	0xYF F000	QADC64 Module Configuration Register (QADC64MCR) See Table 8-7 for bit descriptions.	
T[2]	0xYF F002	QADC64 Test Register (QADC64TEST)	
S	0xYF F004	Interrupt Register (QADC64INT) See Table 8-8 for bit descriptions.	
S/U[3]	0xYF F006	Port A Data (PORTQA) See Table 8-10 for bit descriptions.	Port B Data (PORTQB)
S/U	0xYF F008	Port A Data Direction Register (DDRQA) See Table 8-10 for bit descriptions.	
S/U	0xYF F00A	QADC64 Control Register 0 (QACR0) See Table 8-11 for bit descriptions.	
S/U	0xYF F00C	QADC64 Control Register 1 (QACR1) See Table 8-12 for bit descriptions.	
S/U	0xYF F00E	QADC64 Control Register 2 (QACR2) See Table 8-14 for bit descriptions.	
S/U	0xYF F010	QADC64 Status Register 0 (QASR0) See Table 8-16 for bit descriptions.	
S/U	0xYF F012	QADC64 Status Register 1 (QASR1) See Table 8-16 for bit descriptions.	
—	0xYF F014 – 0xYF F3FE	Reserved	
S/U	0xYF F300 — 0xYF F37F	Conversion Command Word (CCW) Table See Table 8-19 for bit descriptions.	
S/U	0xYF F380 – 0xYF F3FE	Result Word Table Right-Justified, Unsigned Result Register (RJURR) See 8.12.11 Result Word Table for bit descriptions.	
S/U	0xYF F300 – 0xYF F37E	Result Word Table Left-Justified, Signed Result Register (LJSRR) See 8.12.11 Result Word Table for bit descriptions.	
S/U	0xYF F380 – 0xYF F3FE	Result Word Table Left-Justified, Unsigned Result Register (LJURR) See 8.12.11 Result Word Table for bit descriptions.	

NOTES:
1. S = Supervisor only
2. Access is restricted to supervisor only and factory test mode only.
3. S/U = Unrestricted or supervisor depending on the state of the SUPV bit in the QADC64MCR.

The QADC64 has two global registers for configuring module operation: the module configuration register (QADC64MCR) and the interrupt register (QADC64INT). The global registers are always defined to be in supervisor data space. The CPU allows software to establish the global registers in supervisor data space and the remaining registers and tables in user space.

All QADC64 analog channel/port pins that are not used for analog input channels can be used as digital port pins. Port values are read/written by accessing the port A and B data registers (PORTQA and PORTQB). Port A pins are specified as inputs or outputs by programming the port data direction register (DDRQA). Port B is an input only port.

8.12.1 QADC64 Module Configuration Register

QADC64MCR — QADC64 Module Configuration Register **0xYF F000**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
STOP	FRZ	RESERVED						SUPV	RESERVED			IARB			
RESET:															
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 8-7 QADC64MCR Bit Settings

Bit(s)	Name	Description
15	STOP	Low-power stop mode enable. When the STOP bit is set, the clock signal to the QADC64 is disabled, effectively turning off the analog circuitry. 0 = Enable QADC64 clock. 1 = Disable QADC64 clock.
14	FRZ	FREEZE assertion response. The FRZ bit determines whether or not the QADC64 responds to assertion of the IMB3 FREEZE signal. 0 = QADC64 ignores the IMB3 FREEZE signal. 1 = QADC64 finishes any current conversion, then freezes.
13:8	—	Reserved
7	SUPV	Supervisor/unrestricted data space. The SUPV bit designates the assignable space as supervisor or unrestricted. 0 = Only the module configuration register, test register, and interrupt register are designated as supervisor-only data space. Access to all other locations is unrestricted. 1 = All QADC64 registers and tables are designated as supervisor-only data space.
6:4	—	Reserved
3:0	IARB	Interrupt arbitration number. IARB determines QADC64 interrupt arbitration priority. An IARB field can be assigned a value from 0b0001 (lowest priority) to 0b1111 (highest value). Note that the logic associated with the IARB field is implemented for bus masters with interrupt acknowledge cycles (IACK).

8.12.2 QADC64 Interrupt Register

QADC64INT specifies the priority level of QADC64 interrupt requests and the vector provided. The interrupt level for queue 1 and queue 2 may be different. The interrupt register is read/write accessible in supervisor data space only. The implemented interrupt register fields can be read and written, reserved bits read zero and writes have no effect. They are typically written once when the software initializes the QADC, and not changed afterwards.

QADC64INT — QADC64 Interrupt Register **0xYF F004**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED	IRL1			RESERVED	IRL2			IVB							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-8 QADC64INT Bit Settings

Bit(s)	Name	Description
15	—	Reserved
14:12	IRL1	Interrupt level for queue 1. A value of 00000 provides an interrupt level of 0; 0b111 provides a level interrupt. All interrupts are presented on the IMB3. Interrupt level priority software determines which level has the highest priority request.
11	—	Reserved
10:8	IRL2	Interrupt level for queue 2. A value of 00000 provides an interrupt level of 0; 0b111 provides a level interrupt. All interrupts are presented on the IMB3. Interrupt level priority software determines which level has the highest priority request.
7:0	IVB	Interrupt vector base. Initialization software inputs the upper six IVB bits in the interrupt register. During interrupt arbitration, the vector provided to the bus master by the QADC is made up of the upper six IVB bits , plus two low-order bits provided to the QADC to identify one of four QADC interrupt requests. The interrupt vector number is independent of the interrupt level and the interrupt arbitration number. A 0x0F vector number corresponds to the uninitialized interrupt vector. After reset, the lower byte of the interrupt register reads as 0x0F. Once the IVB field is written, the two least significant bits always read as zeros.

8.12.3 Port A/B Data Register

QADC64 ports A and B are accessed through two 8-bit port data registers (PORTQA and PORTQB).

PORTQA — Port QA Data Register **0xYF F006**
PORTQB — Port QB Data Register

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
PQA7	PQA6	PQA5	PQA4	PQA3	PQA2	PQA1	PQA0	PQB7	PQB6	PQB5	PQB4	PQB3	PQB2	PQB1	PQB0
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
ANALOG CHANNEL:															
AN59	AN58	AN57	AN56	AN55	AN54	AN53	AN52	AN51	AN50	AN49	AN48	AN3	AN2	AN1	AN0
MULTIPLEXED ADDRESS OUTPUTS:															
					MA2	MA1	MA0								
MULTIPLEXED ANALOG INPUTS:															
												ANz	ANy	ANx	ANw

Table 8-9 PORTQA, PORTQB Bit Settings

Bit(s)	Name	Description
15:8	PQA[0:7]	Port A pins are referred to as PQA when used as an 8-bit input/output port. Port A can also be used for analog inputs (AN[59:52]), and external multiplexer address outputs (MA[2:0]).
7:0	PQB[0:7]	Port B pins are referred to as PQB when used as an 8-bit input only port. Port B can also be used for non-multiplexed (AN[51:48])/AN[3:0]) and multiplexed (ANz, ANy, ANx, ANw) analog inputs.

8.12.4 Port Data Direction Register

DDRQA — Port QA Data Direction Register **0xYF F008**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
DDQA7	DDQA6	DDQA5	DDQA4	DDQA3	DDQA2	DDQA1	DDQA0	RESERVED							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-10 DDRQA Bit Settings

Bit(s)	Name	Description
15:8	DDQA[7:0]	Bits in this register control the direction of the port QA pin drivers when pins are configured for I/O. Setting a bit configures the corresponding pin as an output; clearing a bit configures the corresponding pin as an input. This register can be read or written at any time.
7:0	—	Reserved

8.12.5 QADC64 Control Register 0 (QACR0)

Control register 0 establishes the QCLK with prescaler parameter fields and defines whether external multiplexing is enabled. All of the implemented control register fields can be read or written, reserved fields read zero and writes have no effect. They are typically written once when the software initializes the QADC64, and not changed afterwards.

QACR0 — QADC64 Control Register 0 **0xYF F00A**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MUX	RESERVED		TRG	RESERVED			PSH					PSA	PSL		
RESET:															
0	0	0	0	0	0	0	0	1	0	1	1	0	1	1	1

Table 8-11 QACR0 Bit Settings

Bit(s)	Name	Description
15	MUX	Externally multiplexed mode. The MUX bit configures the QADC64 for externally multiplexed mode, which affects the interpretation of the channel numbers and forces the MA[2:0] pins to be outputs. 0 = Internally multiplexed, 16 possible channels. AMUX is disabled. 1 = Externally multiplexed, 41 possible channels. This enables the on-chip AMUX.
14:13	—	Reserved
12	TRG	Trigger assignment. TRG allows the software to assign the ETRIG[2:1] pins to queue 1 and queue 2. 0 = ETRIG1 triggers queue 1; ETRIG2 triggers queue 2 1 = ETRIG1 triggers queue 2; ETRIG2 triggers queue 1
11:9	—	Reserved
8:4	PSH	Prescaler clock high time. The PSH field selects the QCLK high time in the prescaler. PSH value plus 1 represents the high time in system clocks
3	PSA	Note that this bit location is maintained for software compatibility with previous versions of the QADC64. It serves no functional benefit in the MC68377 and is not operational.
2:0	PSL	Prescaler clock low time. The PSL field selects the QCLK low time in the prescaler. PSL value plus 1 represents the low time in system clocks

8.12.6 QADC64 Control Register 1 (QACR1)

Control register 1 is the mode control register for the operation of queue 1. The applications software defines the queue operating mode for the queue, and may enable a completion and/or pause interrupt. All of the control register fields are read/write data. However, the SSE1 bit always reads as zero unless the test mode is enabled. Most of the bits are typically written once when the software initializes the QADC64 and not changed afterwards.

QACR1 — Control Register 1 **0xYF F00C**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CIE1	PIE1	SSE1	MQ1					RESERVED							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-12 QACR1 Bit Settings

Bit(s)	Name	Description
15	CIE1	Queue 1 completion interrupt enable. CIE1 enables completion interrupts for queue 1. The interrupt request is generated when the conversion is complete for the last CCW in queue 1. 0 = Queue 1 completion interrupts disabled. 1 = Generate an interrupt request after completing the last CCW in queue 1.
14	PIE1	Queue 1 pause interrupt enable. PIE1 enables pause interrupts for queue 1. The interrupt request is generated when the conversion is complete for a CCW that has the pause bit set. 0 = Queue 1 pause interrupts disabled. 1 = Generate an interrupt request after completing a CCW in queue 1 which has the pause bit set.
13	SSE1	Queue 1 single-scan enable. SSE1 enables a single-scan of queue 1 after a trigger event occurs. The SSE1 bit may be set to a one during the same write cycle that sets the MQ1 bits for the single-scan queue operating mode. The single-scan enable bit can be written as a one or a zero, but is always read as a zero. The SSE1 bit allows a trigger event to initiate queue execution for any single-scan operation on queue 1. The QADC64 clears SSE1 when the single-scan is complete.
12:8	MQ1	Queue 1 operating mode. The MQ1 field selects the queue operating mode for queue 1. **Table 8-13** shows the different queue 1 operating modes.
7:0	—	Reserved

Table 8-13 Queue 1 Operating Modes

MQ1	Operating Modes
00000	Disabled mode, conversions do not occur
00001	Software triggered single-scan mode (started with SSE1)
00010	External trigger rising edge single-scan mode
00011	External trigger falling edge single-scan mode
00100	Interval timer single-scan mode: time = QCLK period x 2^7
00101	Interval timer single-scan mode: time = QCLK period x 2^8
00110	Interval timer single-scan mode: time = QCLK period x 2^9
00111	Interval timer single-scan mode: time = QCLK period x 2^{10}
01000	Interval timer single-scan mode: time = QCLK period x 2^{11}
01001	Interval timer single-scan mode: time = QCLK period x 2^{12}
01010	Interval timer single-scan mode: time = QCLK period x 2^{13}
01011	Interval timer single-scan mode: time = QCLK period x 2^{14}
01100	Interval timer single-scan mode: time = QCLK period x 2^{15}
01101	Interval timer single-scan mode: time = QCLK period x 2^{16}
01110	Interval timer single-scan mode: time = QCLK period x 2^{17}
01111	External gated single-scan mode (started with SSE1)
10000	Reserved mode
10001	Software triggered continuous-scan mode
10010	External trigger rising edge continuous-scan mode
10011	External trigger falling edge continuous-scan mode
10100	Periodic timer continuous-scan mode: time = QCLK period x 2^7
10101	Periodic timer continuous-scan mode: time = QCLK period x 2^8
10110	Periodic timer continuous-scan mode: time = QCLK period x 2^9
10111	Periodic timer continuous-scan mode: time = QCLK period x 2^{10}
11000	Periodic timer continuous-scan mode: time = QCLK period x 2^{11}
11001	Periodic timer continuous-scan mode: time = QCLK period x 2^{12}
11010	Periodic timer continuous-scan mode: time = QCLK period x 2^{13}
11011	Periodic timer continuous-scan mode: time = QCLK period x 2^{14}
11100	Periodic timer continuous-scan mode: time = QCLK period x 2^{15}
11101	Periodic timer continuous-scan mode: time = QCLK period x 2^{16}
11110	Periodic timer continuous-scan mode: time = QCLK period x 2^{17}
11111	External gated continuous-scan mode

8.12.7 QADC64 Control Register 2 (QACR2)

Control register 2 is the mode control register for the operation of queue 2. Software specifies the queue operating mode of queue 2, and may enable a completion and/or a pause interrupt. All control register fields are read/write data, except the SSE2 bit, which is readable only when the test mode is enabled. Most of the bits are typically written once when the software initializes the QADC64 and not changed afterwards.

QACR2 — Control Register 2 **0xYF F00E**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CIE2	PIE2	SSE2	MQ2					RE-SUME	BQ2						
RESET:															
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Table 8-14 QACR2 Bit Settings

Bit(s)	Name	Description
15	CIE2	Queue 2 completion interrupt enable. CIE2 enables completion interrupts for queue 2. The interrupt request is generated when the conversion is complete for the last CCW in queue 2. 0 = Queue 2 completion interrupts disabled. 1 = Generate an interrupt request after completing the last CCW in queue 2.
14	PIE2	Queue 2 pause interrupt enable. PIE2 enables pause interrupts for queue 2. The interrupt request is generated when the conversion is complete for a CCW that has the pause bit set. 0 = Queue 2 pause interrupts disabled. 1 = Generate an interrupt request after completing a CCW in queue 2 which has the pause bit set.
13	SSE2	Queue 2 single-scan enable bit. SSE2 enables a single-scan of queue 2 after a trigger event occurs. The SSE2 bit may be set to a one during the same write cycle that sets the MQ2 bits for the single-scan queue operating mode. The single-scan enable bit can be written as a one or a zero, but is always read as a zero. The SSE2 bit allows a trigger event to initiate queue execution for any single-scan operation on queue 2. The QADC64 clears SSE2 when the single-scan is complete.
12:8	MQ2	Queue 2 operating mode. The MQ2 field selects the queue operating mode for queue 2. **Table 8-15** shows the bits in the MQ2 field which enable different queue 2 operating modes.
7	RESUME	Queue 2 resume. RESUME selects the resumption point after queue 2 is suspended by queue 1. If RESUME is changed during execution of queue 2, the change is not recognized until an end-of-queue condition is reached, or the queue operating mode of queue 2 is changed. 0 = After suspension, begin execution with the first CCW in queue 2 or the current subqueue. 1 = After suspension, begin execution with the aborted CCW in queue 2.
6:0	BQ2	Beginning of queue 2. The BQ2 field indicates the location in the CCW table where queue 2 begins. The BQ2 field also indicates the end-of-queue 1 and thus creates an end-of-queue condition for queue 1. Setting BQ2 to any value ≥ 64 (1000000) allows the entire RAM space for queue 1 CCW's.

Table 8-15 Queue 2 Operating Modes

MQ2	Operating Modes
00000	Disabled mode, conversions do not occur
00001	Software triggered single-scan mode (started with SSE2)
00010	External trigger rising edge single-scan mode
00011	External trigger falling edge single-scan mode
00100	Interval timer single-scan mode: interval = QCLK period x 2^7
00101	Interval timer single-scan mode: interval = QCLK period x 2^8
00110	Interval timer single-scan mode: interval = QCLK period x 2^9
00111	Interval timer single-scan mode: interval = QCLK period x 2^{10}
01000	Interval timer single-scan mode: interval = QCLK period x 2^{11}
01001	Interval timer single-scan mode: interval = QCLK period x 2^{12}
01010	Interval timer single-scan mode: interval = QCLK period x 2^{13}
01011	Interval timer single-scan mode: interval = QCLK period x 2^{14}
01100	Interval timer single-scan mode: interval = QCLK period x 2^{15}
01101	Interval timer single-scan mode: interval = QCLK period x 2^{16}
01110	Interval timer single-scan mode: interval = QCLK period x 2^{17}
01111	Reserved mode
10000	Reserved mode
10001	Software triggered continuous-scan mode (started with SSE2)
10010	External trigger rising edge continuous-scan mode
10011	External trigger falling edge continuous-scan mode
10100	Periodic timer continuous-scan mode: period = QCLK period x 2^7
10101	Periodic timer continuous-scan mode: period = QCLK period x 2^8
10110	Periodic timer continuous-scan mode: period = QCLK period x 2^9
10111	Periodic timer continuous-scan mode: period = QCLK period x 2^{10}
11000	Periodic timer continuous-scan mode: period = QCLK period x 2^{11}
11001	Periodic timer continuous-scan mode: period = QCLK period x 2^{12}
11010	Periodic timer continuous-scan mode: period = QCLK period x 2^{13}
11011	Periodic timer continuous-scan mode: period = QCLK period x 2^{14}
11100	Periodic timer continuous-scan mode: period = QCLK period x 2^{15}
11101	Periodic timer continuous-scan mode: period = QCLK period x 2^{16}
11110	Periodic timer continuous-scan mode: period = QCLK period x 2^{17}
11111	Reserved mode

8.12.8 QADC64 Status Register 0 (QASR0)

QASR0 contains information about the state of each queue and the current A/D conversion. Except for the four flag bits (CF1, PF1, CF2, and PF2) and the two trigger overrun bits (TOR1 and TOR2), all of the status register fields contain read-only data. The four flag bits and the two trigger overrun bits are cleared by writing a zero to the bit after the bit was previously read as a one.

QASR0 — QADC64 Status Register 0xYF F010

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CF1	PF1	CF2	PF2	TOR1	TOR2	QS				CWP					
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-16 QASR0 Bit Settings

Bit(s)	Name	Description
15	CF1	Queue 1 completion flag. CF1 indicates that a queue 1 scan has been completed. CF1 is set by the QADC64 when the conversion is complete for the last CCW in queue 1, and the result is stored in the result table. 0 = Queue 1 scan is not complete. 1 = Queue 1 scan is complete.
14	PF1	Queue 1 pause flag. PF1 indicates that a queue 1 scan has reached a pause. PF1 is set by the QADC64 when the current queue 1 CCW has the pause bit set, the selected input channel has been converted, and the result has been stored in the result table. 0 = Queue 1 has not reached a pause. 1 = Queue 1 has reached a pause.
13	CF2	Queue 2 completion flag. CF2 indicates that a queue 2 scan has been completed. CF2 is set by the QADC64 when the conversion is complete for the last CCW in queue 2, and the result is stored in the result table. 0 = Queue 2 scan is not complete. 1 = Queue 2 scan is complete.
12	PF2	Queue 2 pause flag. PF2 indicates that a queue 2 scan has reached a pause. PF2 is set by the QADC64 when the current queue 2 CCW has the pause bit set, the selected input channel has been converted, and the result has been stored in the result table. 0 = Queue 2 has not reached a pause. 1 = Queue 2 has reached a pause.
11	TOR1	Queue 1 trigger overrun. TOR1 indicates that an unexpected queue 1 trigger event has occurred. TOR1 can be set only while queue 1 is active. A trigger event generated by a transition on ETRIG1/ETRIG2 may be recorded as a trigger overrun. TOR1 can only be set when using an external trigger mode. TOR1 cannot occur when the software initiated single-scan mode or the software initiated continuous-scan mode is selected. 0 = No unexpected queue 1 trigger events have occurred. 1 = At least one unexpected queue 1 trigger event has occurred.
10	TOR2	Queue 2 trigger overrun. TOR2 indicates that an unexpected queue 2 trigger event has occurred. TOR2 can be set when queue 2 is in the active, suspended, and trigger pending states. A trigger event generated by a transition depending on the value of TRG in QACR or ETRIG1/ETRIG2 or by the periodic/interval timer may be recorded as a trigger overrun. TOR2 can only be set when using an external trigger mode or a periodic/interval timer mode. Trigger overruns cannot occur when the software initiated single-scan mode and the software initiated continuous-scan mode are selected. 0 = No unexpected queue 2 trigger events have occurred. 1 = At least one unexpected queue 2 trigger event has occurred.
9:6	QS	Queue status. This 4-bit read-only field indicates the current condition of queue 1 and queue 2. QS[0:1] are associated with queue 1, and QS[2:3] are associated with queue 2. Since the queue priority scheme interlinks the operation of queue 1 and queue 2, the status bits should be considered as one 4-bit field. Table 8-17 shows the bit encodings of the QS field.
5:0	CWP	Command word pointer. CWP indicates which CCW is executing at present, or was last completed. The CWP is a read-only field; writes to it have no effect.

Table 8-17 Queue Status

QS	Description
0000	Queue 1 idle, queue 2 idle
0001	Queue 1 idle, queue 2 paused
0010	Queue 1 idle, queue 2 active
0011	Queue 1 idle, queue 2 trigger pending
0100	Queue 1 paused, queue 2 idle
0101	Queue 1 paused, queue 2 paused
0110	Queue 1 paused, queue 2 active
0111	Queue 1 paused, queue 2 trigger pending
1000	Queue 1 active, queue 2 idle
1001	Queue 1 active, queue 2 paused
1010	Queue 1 active, queue 2 suspended
1011	Queue 1 active, queue 2 trigger pending
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

8.12.9 QADC64 Status Register 1 (QASR1)

The QASR1 contains two fields: command word pointers for queue 1 and queue 2.

QASR1 — Status Register 1 **0xYF F012**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED		CWPQ1						RESERVED		CWPQ2					
RESET:															
0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1

Table 8-18 QASR1 Bit Settings

Bit(s)	Name	Description
15:14	—	Reserved
13:8	CWPQ1	Command word pointer for queue 1. This field is a software read-only field, and write operations have no effect. CWPQ1 allows software to read the last executed CCW in queue 1, regardless which queue is active. The CWPQ1 field is a CCW word pointer with a valid range of 0 to 63. In contrast to CWP, CPWQ1 is updated when the conversion result is written. When the QADC64 finishes a conversion in queue 1, both the result register is written and the CWPQ1 are updated. Finally, when queue 1 operation is terminated after a CCW is read that is defined as BQ2, CWP points to BQ2 while CWPQ1 points to the last CCW queue 1. During the stop mode, the CWPQ1 is reset to 63, since the control registers and the analog logic are reset. When the freeze mode is entered, the CWPQ1 is unchanged; it points to the last executed CCW in queue 1.
7:6	—	Reserved
5:0	CWPQ2	Command word pointer for queue 2. This field is a software read-only field, and write operations have no effect. CWPQ2 allows software to read the last executed CCW in queue 2, regardless which queue is active. The CWPQ2 field is a CCW word pointer with a valid range of 0 to 63. In contrast to CWP, CPWQ2 is updated when the conversion result is written. When the QADC64 finishes a conversion in queue 2, both the result register is written and the CWPQ2 are updated. During the stop mode, the CWPQ2 is reset to 63, since the control registers and the analog logic are reset. When the freeze mode is entered, the CWP is unchanged; it points to the last executed CCW in queue 2.

8.12.10 Conversion Command Word Table

The CCW table is a RAM, 64 words long and 10 bits wide, which can be programmed by the software to request conversions of one or more analog input channels. The entries in the CCW table are 10-bit conversion command words. The CCW table is written by software and is not modified by the QADC64. Each CCW requests the conversion of an analog channel to a digital result. The CCW specifies the analog channel number, the input sample time, and whether the queue is to pause after the current CCW.

The ten implemented bits of the CCW word are read/write data. They may be written when the software initializes the QADC64. Unimplemented bits are read as zeros, and write operations have no effect. Each location in the CCW table corresponds to a location in the result word table. When a conversion is completed for a CCW entry, the 10-bit result is written in the corresponding result word entry. The QADC64 provides 64 CCW table entries.

The beginning of queue 1 is always the first location in the CCW table. The first location of queue 2 is specified by the beginning of queue 2 pointer (BQ2) in QACR2. To dedicate the entire CCW table to queue 1, software must do the following:

- Program queue 2 to be in the disabled mode, and
- Program the beginning of BQ2 to ≥ 64.

To dedicate the entire CCW table to queue 2, software must do the following:

- Program queue 1 to be in the disabled mode
- Program BQ2 to be the first location in the CCW table.

Figure 8-12 illustrates the operation of the queue structure.

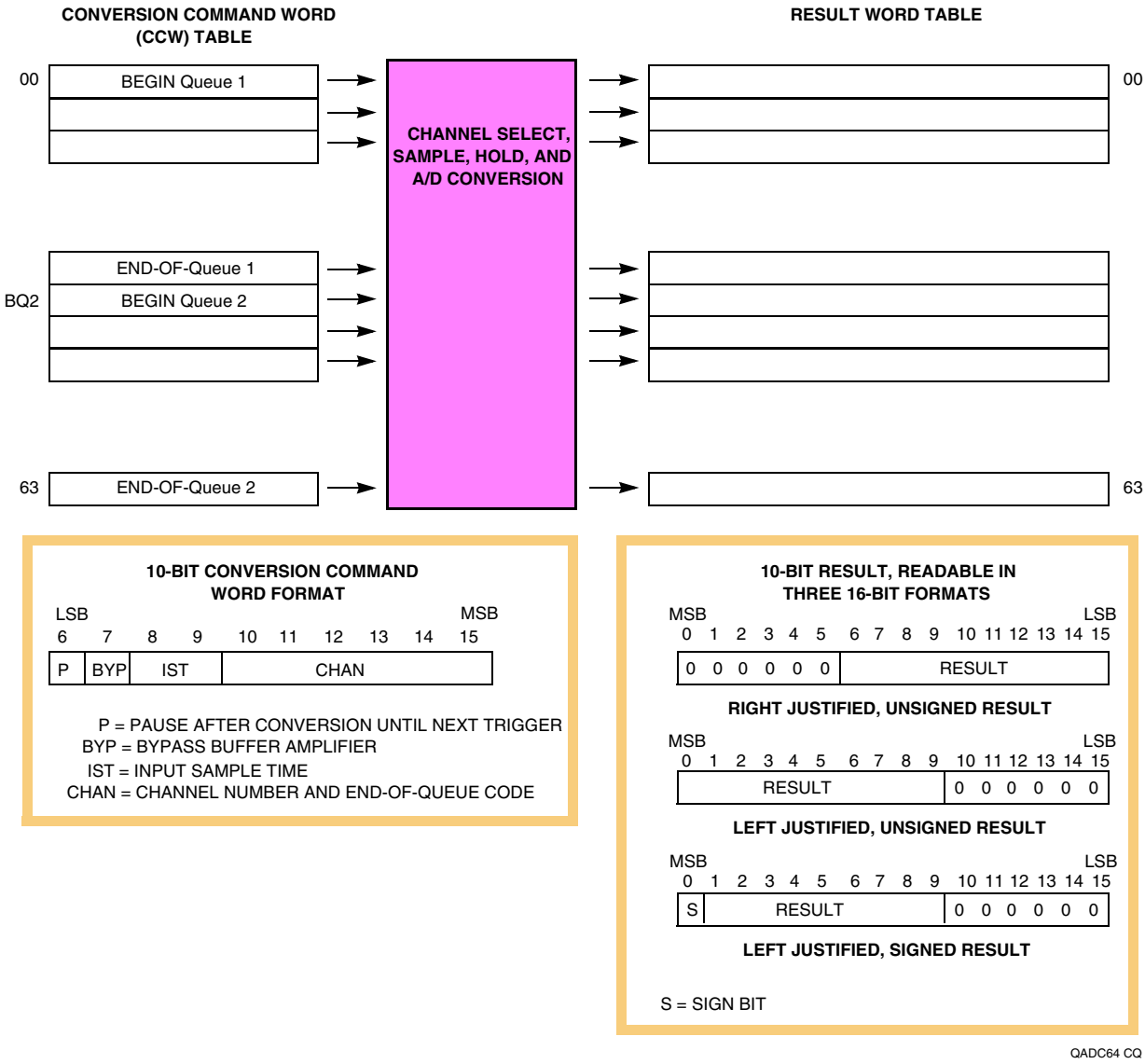


Figure 8-12 QADC64 Conversion Queue Operation

To prepare the QADC64 for a scan sequence, the software writes to the CCW table to specify the desired channel conversions. The software also establishes the criteria for initiating the queue execution by programming the queue operating mode. The queue operating mode determines what type of trigger event causes queue execution to begin. “Trigger event” refers to any of the ways to cause the QADC64 to begin executing the CCWs in a queue or subqueue. An external trigger is only one of the possible trigger events.

A scan sequence may be initiated by the following:

- A software command

- Expiration of the periodic/interval timer
- External trigger signal
- External gated signal (queue 1 only)

The software also specifies whether the QADC64 is to perform a single pass through the queue or is to scan continuously. When a single-scan mode is selected, the software selects the queue operating mode and sets the single-scan enable bit. When a continuous-scan mode is selected, the queue remains active in the selected queue operating mode after the QADC64 completes each queue scan sequence.

During queue execution, the QADC64 reads each CCW from the active queue and executes conversions in three stages:

- Initial sample
- Final sample
- Resolution

During initial sample, a buffered version of the selected input channel is connected to the sample capacitor at the output of the sample buffer amplifier.

During the final sample period, the sample buffer amplifier is bypassed, and the multiplexer input charges the sample capacitor directly. Each CCW specifies a final input sample time of 2, 4, 8, or 16 QCLK cycles. When an analog-to-digital conversion is complete, the result is written to the corresponding location in the result word table. The QADC64 continues to sequentially execute each CCW in the queue until the end of the queue is detected or a pause bit is found in a CCW.

When the pause bit is set in the current CCW, the QADC64 stops execution of the queue until a new trigger event occurs. The pause status flag bit is set, which may cause an interrupt to notify the software that the queue has reached the pause state. After the trigger event occurs, the paused state ends and the QADC64 continues to execute each CCW in the queue until another pause is encountered or the end of the queue is detected.

The following indicate the end-of-queue condition:

- The CCW channel field is programmed with 63 (0x3F) to specify the end of the queue.
- The end-of-queue 1 is implied by the beginning of queue 2, which is specified in the BQ2 field in QACR2.
- The physical end of the queue RAM space defines the end of either queue.

When any of the end-of-queue conditions is recognized, a queue completion flag is set, and if enabled, an interrupt is issued to the software. The following situations prematurely terminate queue execution:

- Since queue 1 is higher in priority than queue 2, when a trigger event occurs on queue 1 during queue 2 execution, the execution of queue 2 is suspended by aborting the execution of the CCW in progress, and the queue 1 execution begins. When queue 1 execution is completed, queue 2 conversions restart with the first CCW entry in queue 2 or the first CCW of the queue 2 subqueue being exe-

cuted when queue 2 was suspended. Alternately, conversions can restart with the aborted queue 2 CCW entry. The RESUME bit in QACR2 allows the software to select where queue 2 begins after suspension. By choosing to re-execute all of the suspended queue 2 queue and subqueue CCWs, all of the samples are guaranteed to have been taken during the same scan pass. However, a high trigger event rate for queue 1 can prohibit the completion of queue 2. If this occurs, the software may choose to begin execution of queue 2 with the aborted CCW entry.

- Software can change the queue operating mode to disabled mode. Any conversion in progress for that queue is aborted. Putting a queue into the disabled mode does not power down the converter.
- Software can change the queue operating mode to another valid mode. Any conversion in progress for that queue is aborted. The queue restarts at the beginning of the queue, once an appropriate trigger event occurs.
- For low power operation, software can set the stop mode bit to prepare the module for a loss of clocks. The QADC64 aborts any conversion in progress when the stop mode is entered.
- When the freeze enable bit is set by software and the IMB internal FREEZE line is asserted, the QADC64 freezes at the end of the conversion in progress. When internal FREEZE is negated, the QADC64 resumes queue execution beginning with the next CCW entry.

CCW — Conversion Command Word Table **0xYF F300 – 0xYF F37F**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED						P	BYP	IST		CHAN					
RESET:															
0	0	0	0	0	0	U	U	U	U	U	U	U	U	U	U

Table 8-19 CCW Bit Settings

Bit(s)	Name	Description
15:10	—	Reserved
9	P	Pause. The pause bit allows the creation of sub-queues within queue 1 and queue 2. The QADC64 performs the conversion specified by the CCW with the pause bit set, and then the queue enters the pause state. Another trigger event causes execution to continue from the pause to the next CCW. 0 = Do not enter the pause state after execution of the current CCW. 1 = Enter the pause state after execution of the current CCW.
8	BYP	Sample amplifier bypass. Setting BYP enables the amplifier bypass mode for a conversion, and subsequently changes the timing. Refer to **8.9.1.1 Amplifier Bypass Mode Conversion Timing** for more information. 0 = Amplifier bypass mode disabled. 1 = Amplifier bypass mode enabled.
7:6	IST	Input sample time. The IST field specifies the length of the sample window. Longer sample times permit more accurate A/D conversions of signals with higher source impedances, especially if BYP=1. 00 = QCKL period x 2 01 = QCKL period x 4 10 = QCKL period x 8 11 = QCKL period x 16
5:0	CHAN	Channel number. The CHAN field selects the input channel number corresponding to the analog input pin to be sampled and converted. The analog input pin channel number assignments and the pin definitions vary depending on whether the QADC64 is operating in multiplexed or non-multiplexed mode. The queue scan mechanism sees no distinction between an internally or externally multiplexed analog input. If CHAN specifies a reserved channel number (channels 32 to 47) or an invalid channel number (channels 4 to 31 in non-multiplexed mode), the low reference level (V_{RL}) is converted. Programming the channel field to channel 63 indicates the end of the queue. Channels 60 to 62 are special internal channels. When one of these channels is selected, the sample amplifier is not used. The value of V_{RL}, V_{RH}, or ($V_{RH} - V_{RL}$)/2 is placed directly into the converter. Programming the input sample time to any value other than two for one of the internal channels has no benefit except to lengthen the overall conversion time. **Table 8-20** shows the channel number assignments for the non-multiplexed mode. **Table 8-21** shows the channel number assignments for the multiplexed mode.

Table 8-20 Non-Multiplexed Channel Assignments and Pin Designations

Non-multiplexed Input Pins				Channel Number in CHAN	
Port Pin Name	Analog Pin Name	Other Functions	Pin Type	Binary	Decimal
PQB0	AN0	—	Input	000000	0
PQB1	AN1	—	Input	000001	1
PQB2	AN2	—	Input	000010	2
PQB3	AN3	—	Input	000011	3
—	—	Invalid	—	000100 to 011111	4 to 31
—	—	Reserved	—	10XXXX	32 to 47
PQB4	AN48	—	Input	110000	48
PQB5	AN49	—	Input	110001	49
PQB6	AN50	—	Input	110010	50
PQB7	AN51	—	Input	110011	51
PQA0	AN52	—	Input/Output	110100	52
PQA1	AN53	—	Input/Output	110101	53
PQA2	AN54	—	Input/Output	110110	54
PQA3	AN55	—	Input/Output	110111	55
PQA4	AN56	—	Input/Output	111000	56
PQA5	AN57	—	Input/Output	111001	57
PQA6	AN58	—	Input/Output	111010	58
PQA7	AN59	—	Input/Output	111011	59
—	V_{RL}	—	Input	111100	60
—	V_{RH}	—	Input	111101	61
—	—	$(V_{RH} - V_{RL})/2$	—	111110	62
—	—	End-of-Queue Code	—	111111	63

Table 8-21 Multiplexed Channel Assignments and Pin Designations

Multiplexed Input Pins				Channel Number in CHAN	
Port Pin Name	Analog Pin Name	Other Functions	Pin Type	Binary	Decimal
PQB0	ANw	—	Input	00xxx0	0 to 14 even
PQB1	ANx	—	Input	00xxx1	1 to 15 odd
PQB2	ANy	—	Input	01xxx0	16 to 30 even
PQB3	ANz	—	Input	01xxx1	17 to 31 odd
—	—	Reserved	—	10xxxx	32 to 47
PQB4	AN48	—	Input	110000	48
PQB5	AN49	—	Input	110001	49
PQB6	AN50	—	Input	110010	50
PQB7	AN51	—	Input	110011	51
PQA0	—	MA0	Input/Output	110100	52
PQA1	—	MA1	Input/Output	110101	53
PQA2	—	MA2	Input/Output	110110	54
PQA3	AN55	—	Input/Output	110111	55
PQA4	AN56	—	Input/Output	111000	56
PQA5	AN57	—	Input/Output	111001	57
PQA6	AN58	—	Input/Output	111010	58
PQA7	AN59	—	Input/Output	111011	59
—	V_{RL}	—	Input	111100	60
—	V_{RH}	—	Input	111101	61
—	—	$(V_{RH} - V_{RL})/2$	—	111110	62
—	—	End-of-Queue Code	—	111111	63

8.12.11 Result Word Table

The result word table is a 64-word long, 10-bit wide RAM. The QADC64 writes a result word after completing an analog conversion specified by the corresponding CCW. The result word table can be read or written, but in normal operation, software reads the result word table to obtain analog conversions from the QADC64. Unimplemented bits are read as zeros, and write operations have no effect.

While there is only one result word table, the data can be accessed in three different alignment formats:

- Right justified, with zeros in the higher order unused bits.
- Left justified, with the most significant bit inverted to form a sign bit, and zeros in the unused lower order bits.
- Left justified, with zeros in the unused lower order bits.

The left justified, signed format corresponds to a half-scale, offset binary, two's complement data format. The data is routed onto the IMB according to the selected format. The address used to access the table determines the data alignment format. All write operations to the result word table are right justified.

RJURR — Right Justified, Unsigned Result Register **0xYF F380 – 0xYF F3FE**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED						RESULT									
RESET:															
0	0	0	0	0	0										

The conversion result is unsigned, right justified data. Unused bits return zero when read.

LJSRR — Left Justified, Signed Result Register **0xYF F300 – 0xYF F37E**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
S[1]	RESULT									RESERVED					
RESET:															
										0	0	0	0	0	0

NOTES:
1. S = Sign bit.

The conversion result is signed, left justified data. Unused bits return zero when read.

LJURR — Left Justified, Unsigned Result Register **0xYF F380 – 0xYF F3FE**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESULT										RESERVED					
RESET:															
										0	0	0	0	0	0

The conversion result is unsigned, left justified data. Unused bits return zero when read.

8.13 Analog Multiplexer Submodule

The analog multiplexer (AMUX) submodule expands the channel capacity of the QADC64 analog-to-digital converter inputs to a maximum of 27 analog channels (using only the on-chip multiplexer, 41 with the addition of off-chip multiplexers). The AMUX does not have an inter-module bus (IMB3) interface; control is through the QADC64.

The AMUX is a “high voltage” device requiring 5 V nominal. There is no on-board charge pump as with the previous UDR implementation of the AMUX. Performance of the AMUX should be superior to that of an external multiplexer because of the greatly reduced parasitic capacitances. In addition, precautions have been taken to insure that the input current requirement is as low as possible.

The architecture and pin naming of the AMUX were modeled after the QADC64 operating in external multiplexer mode. Use of this feature of the QADC64 is described in **8.7 External Multiplexing Operation**, and is illustrated in **Figure 8-13**. When the AMUX is used with the QADC64, it replaces the external multiplexers. The software model is identical and the system performance is improved.

8.13.1 Signal Descriptions

8.13.1.1 External Pins (Connected to Pads)

ANX0–ANX15, analog input pins, extended. These 16 analog input channels are divided into two groups of eight inputs: ANX0–14 even, ANX1–15 odd. Each group is connected to a separate multiplexer within the AMUX. These pads are unique to the AMUX.

V_{DDA}, V_{SSA}, analog power, is shared by the QADC64 and the AMUX.

V_{DD}, V_{SS}, digital power, is shared by the QADC64 and the AMUX.

8.13.1.2 Internal Pins (Connected to QADC64)

See table **Table 8-22** for a summary of this section.

AMMA[2:0]. Multiplexer address inputs. These pins are decoded into eight select lines that steer the four multiplexers within the AMUX in parallel. The final stage selection between the four multiplexers is done within the QADC64. On the MC68377 only two analog multiplexers are selected.

Assuming that these lines change simultaneously, break-before-make switching ensures that on each channel change, all multiplexer outputs are opened before the new one is selected. This prevents two inputs in the same mux from being connected to the mux output at any one time.

AMCHAN[1:0]. Address lines to select between the four multiplexers. This selection appears redundant since the final selection between the four multiplexers is done by the QADC64 module. This redundancy is added to reduce the dynamic charging current required from the input pin. These pins perform a disable function more than a select function. Only one of the four multiplexers can be active at a time. These address lines disable the other three multiplexers so that none of their switches are closed. As the selected multiplexer cycles through its various inputs, the other multiplexers draw no dynamic charging current through their inputs. Without this disable function (as in the external multiplexer case), pins on all four multiplexers would draw current even though only one multiplexer would actually be in use.

Assuming that AMCHAN[1:0] change simultaneously with AMMA[2:0], break before make switching ensures that on each channel change the following occurs:

1. All four multiplexers are disabled
2. The desired input ANX0-15 is selected
3. The appropriate multiplexer is enabled

AMPMUX. When low this signal disables all multiplexer inputs. It is used in conjunction with AMCHAN[1:0] to reduce dynamic charging current into the AMUX analog pins. This signal is normally low when the direct inputs to the QADC64 are selected.

AMADB[1:0]. The two multiplexer outputs from the AMUX.

Table 8-22 AMUX I/O Functionality

AMPMUX	AMCHAN[1:0]	AMMA[2:0]	AMADB3	AMADB2	AMADB1	AMADB0
0	XX	XXX	Z	Z	Z	Z
1	00	000	Z	Z	Z	ANX0
1	00	001	Z	Z	Z	ANX2
1	00	010	Z	Z	Z	ANX4
1	00	011	Z	Z	Z	ANX6
1	00	100	Z	Z	Z	ANX8
1	00	101	Z	Z	Z	ANX10
1	00	110	Z	Z	Z	ANX12
1	00	111	Z	Z	Z	ANX14
1	01	000	Z	Z	ANX1	Z
1	01	001	Z	Z	ANX3	Z
1	01	010	Z	Z	ANX5	Z
1	01	011	Z	Z	ANX7	Z
1	01	100	Z	Z	ANX9	Z
1	01	101	Z	Z	ANX11	Z
1	01	110	Z	Z	ANX13	Z
1	01	111	Z	Z	ANX15	Z

8.13.2 Mixed AMUX/External Multiplexing

The QADC64/AMUX combination uses a mixed internal/external multiplexing. In mixed multiplexing, the AMUX outputs will be tied to port B pads B[1:0]. Note that the on-chip AMUX is susceptible to noise injection on the user board through the pad input + ESD (electro-static discharge) protection circuitry + the substrate (package) capacitance on its inputs. Care should be taken in the layout of any connections to the AN0/ANW/PQB0 and AN1/ANX/PQB1 pins. If the on-chip AMUX is not disabled, these pins could be used for analog inputs. See **8.13.3 Pin Connection and Performance Considerations** for loading implications.

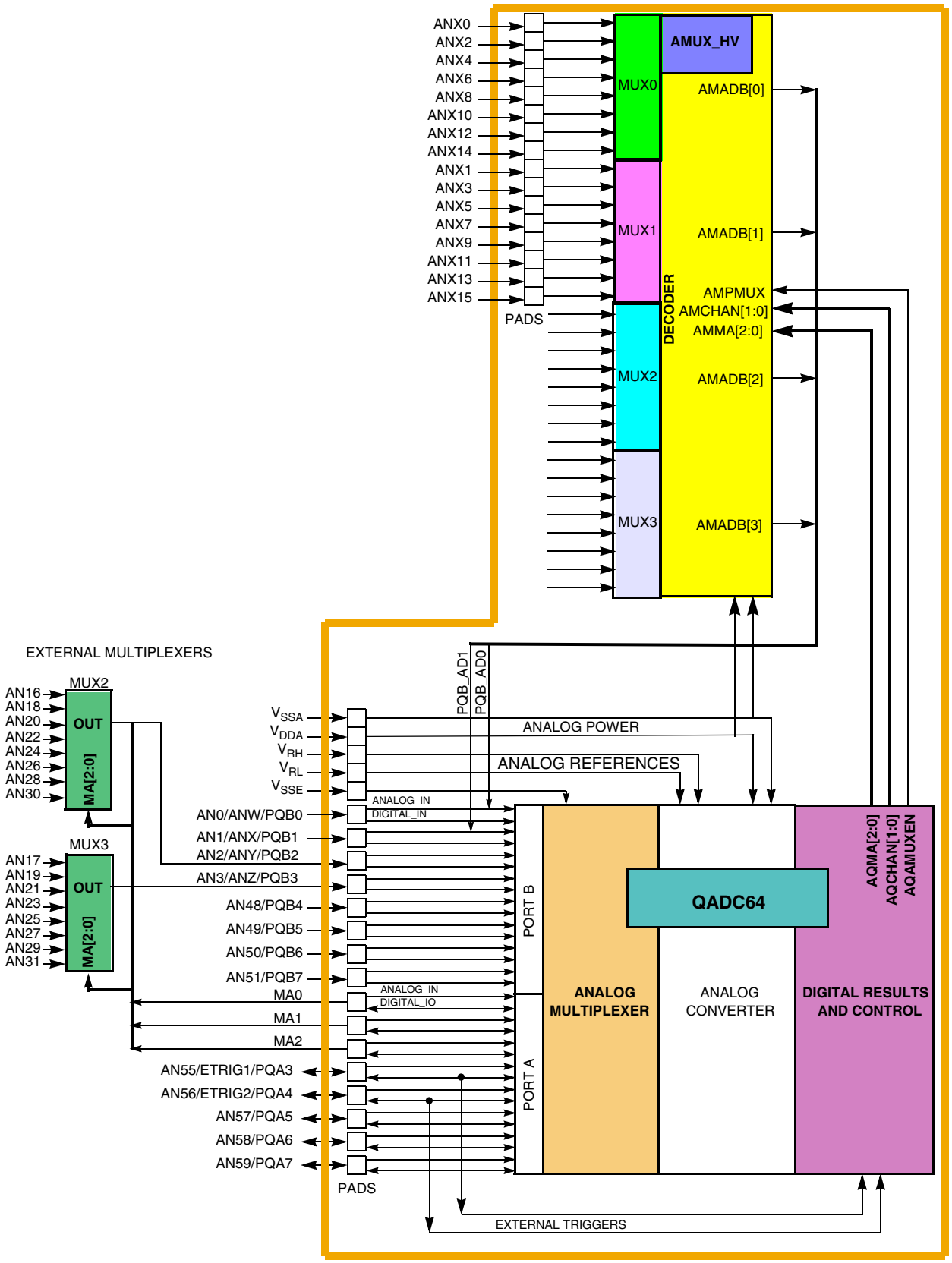


Figure 8-13 AMUX/QADC64 Configured for Mixed Multiplexing

8.13.3 Pin Connection and Performance Considerations

The performance of an ANXx pin is almost the same as that of an ANx input to the QADC64. There are two differences: there is an increased resistance because of the additional multiplexer switch in series, and there is no buffering between channels within the groups of 8. If the increased resistance becomes a problem it can be compensated for by increasing the input sample time (IST[7:6] in the conversion command word table of the QADC64). The lack of buffering between channels causes additional dynamic charging current to flow into the analog pins during successive conversions of different voltages. A schematic of one group of eight inputs which illustrates the cause of this charging current is shown in figure **Figure 8-14**.

If alternating between 2 AMUX channels within the group of eight, one at V_{rh} and the other at V_{rl}, the equivalent capacitance C_{EQ} must be charged/discharged between V_{rh} and V_{rl}. This generates an average current which flows across the sum of the filter, source, and mux resistances. The error due to the charging current is expressed by the following equation:

$$ERROR = I_{AVG} * R_{EQ} = \Delta V * sample\_rate * C_{EQ} * R_{EQ}$$

Where: $C_{EQ} = C_{MUXOUT} + C_P$

$$R_{EQ} = R_{SOURCE} + R_{FILTER} + R_{MUXOUT}$$

$$\Delta V = V_{rh} - V_{rl}$$

For an error equal to 1/2 LSB, the following equation must be met:

$$C_{EQ} * R_{EQ} < 1/(2048*sample\_rate)$$

As an example, if $C_{EQ} = 2.0$ pF and $R_{EQ} = 50K$, the maximum sample_rate is 4.9 KHz.

This charging current is also present on normal QADC64 channels, but to a lesser extent because C_{MUXOUT} and R_{MUXOUT} are both zero.

For a discussion of other pin connection and performance considerations, please refer to **8.3 QADC64 Pin Functions**.

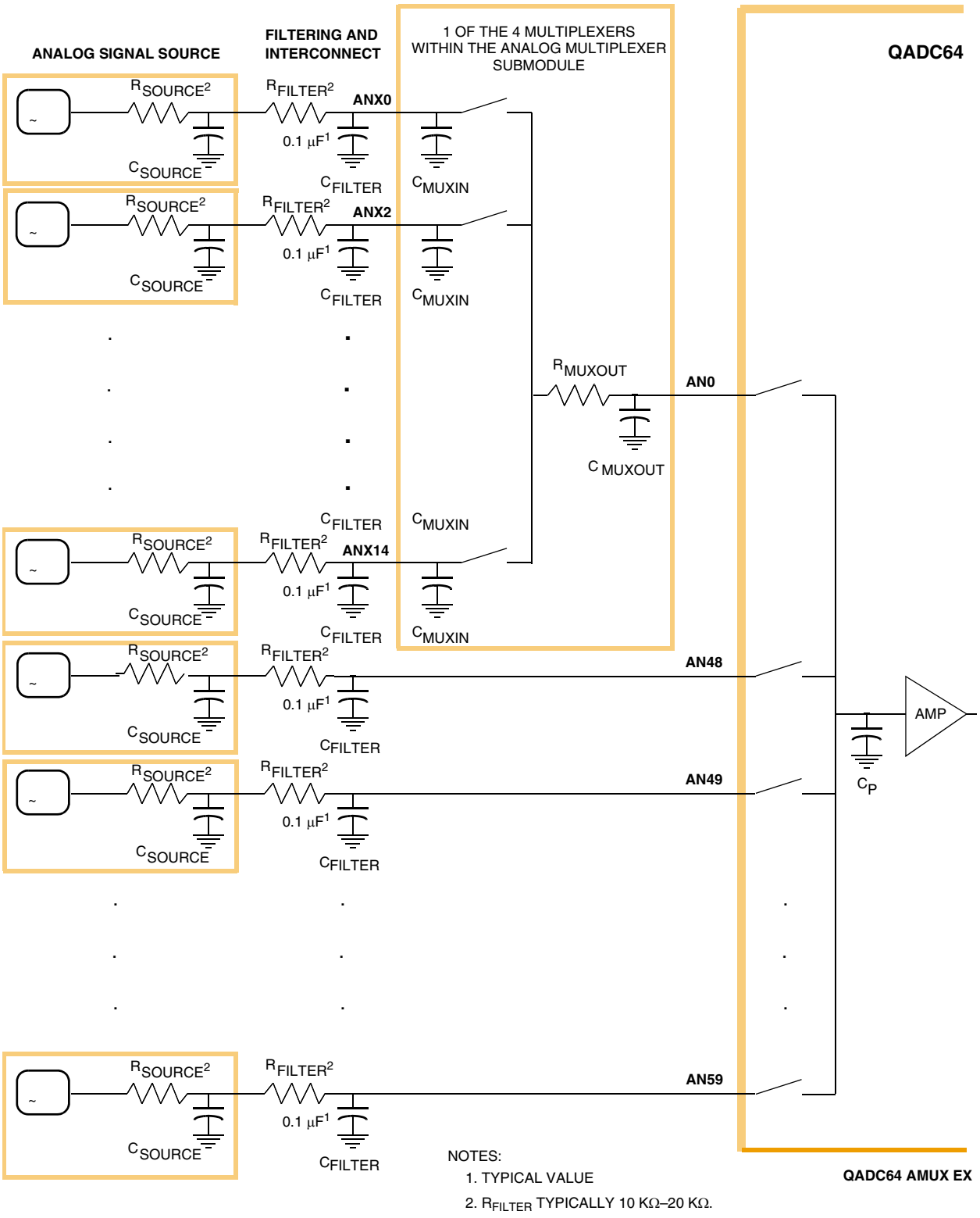


Figure 8-14 Analog Multiplexer Submodule Charging Current Illustration

SECTION 9
CONFIGURABLE TIMER MODULE (CTM9)

9.1 Introduction

The configurable timer module (CTM9) is a family of timer modules for the Motorola modular microcontroller family (MMF), including the MC68300 (CPU32) and the MC68HC16 (CPU16) families of microcontrollers (MCUs). The timer architecture is modular relative to the number of time-bases (counter submodules), channels (action submodules) and other available general purpose functions (real time clock, RAM, I/O ports, etc.) that can be included.

Please refer to the ***CTM Reference Manual*** **(CTMRM/D)** for more information.

9.1.1 CTM9 Configuration

The CTM9 is composed of the following submodules:

- One free-running counter submodule (FCSM).
- Two modulus counter submodule (MCSM).
- Four single action submodule (SASM).
- Four double action submodule (DASM).
- Four dedicated PWM submodule (PWMSM).
- One bus interface unit submodule (BIUSM).
- One counter prescaler submodule (CPSM).

Figure 9-1 and **Table 9-1** show respectively a block diagram and a table representation of the CTM9 configuration.

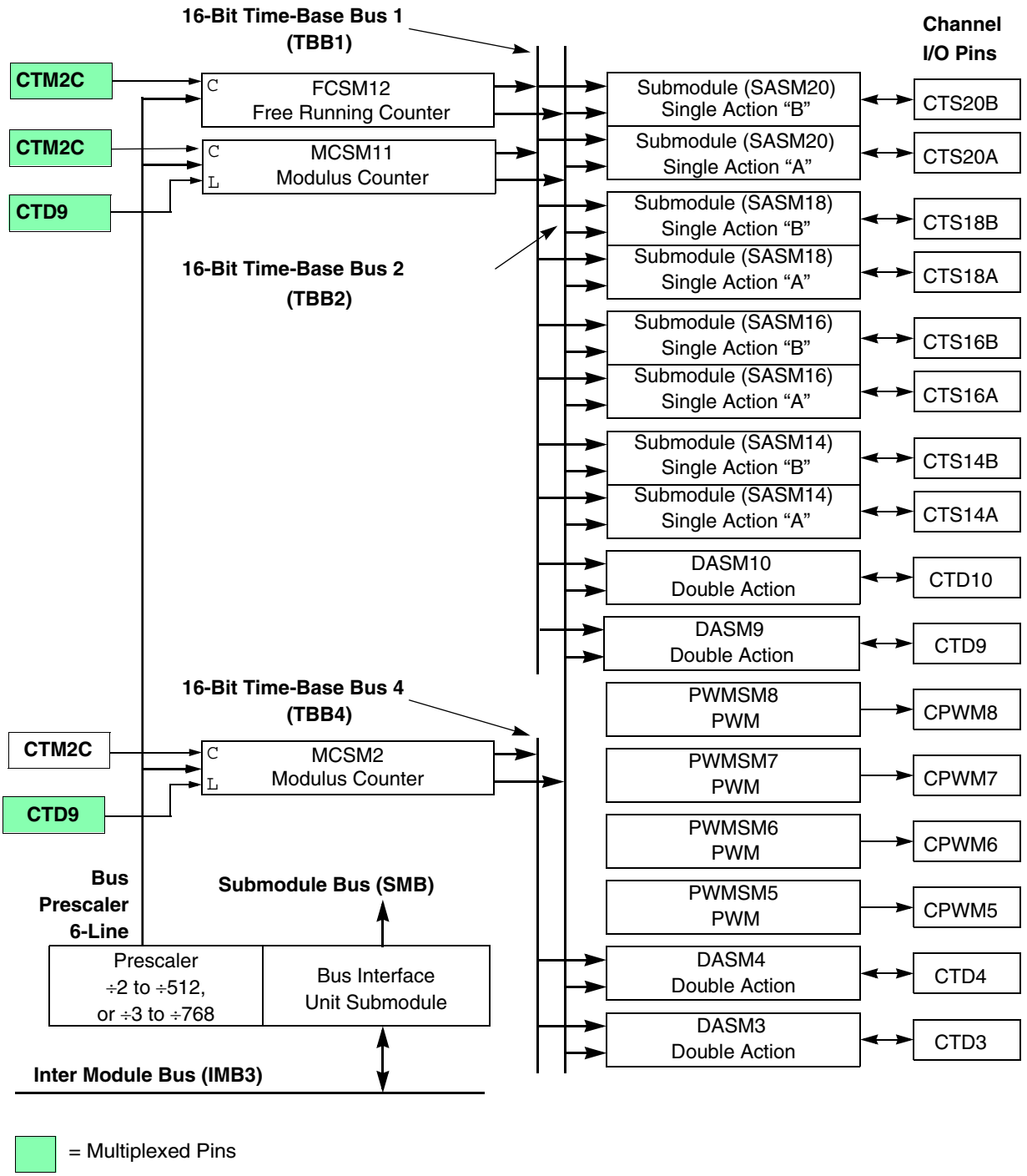


Figure 9-1 Configurable Timer Module, CTM9 Block Diagram

Table 9-1 CTM9 Configuration Description

Submodule Type	Submodule Number	Connected to: tbb1 (A)	tbb2 (B)	tbb3 (B)	tbb4 (A)	Binary Interrupt Vector Number	Submodule Base Address	Pin Function	Input Pin Name	Output Pin Name
BIUSM+CPSM	0						0xYF F700			
MCSM	2[1]	0	1	0	1	0bxx000010[2]	0xYF F710[3]	Clock	CTM2C	
								Load	CTD9	
DASM	3	0	1	0	1	0bxx000011	0xYF F718	Channel	CTD3	CTD3
DASM	4	0	1	0	1	0bxx000100	0xYF F720	Channel	CTD4	CTD4
PWMSM	5					0bxx000101	0xYF F728	Channel		CPWM5
PWMSM	6					0bxx000110	0xYF F730	Channel		CPWM6
PWMSM	7					0bxx000111	0xYF F738	Channel		CPWM7
PWMSM	8					0bxx001000	0xYF F740	Channel		CPWM8
DASM	9	1	1	0	0	0bxx001001	0xYF F748	Channel	CTD9	CTD9
DASM	10	1	1	0	0	0bxx001010	0xYF F750	Channel	CTD10	CTD10
MCSM	11	1	1	0	0	0bxx001011	0xYF F758	Clock	CTM2C	
								Load	CTD9	
FCSM	12	1	1	0	0	0bxx001100	0xYF F760	Clock	CTM2C	
SASM	14	1	1	0	0	0bxx001110	0xYF F770	Channel A	CTS14A	CTS14A
						0bxx001111		Channel B	CTS14B	CTS14B
SASM	16	1	1	0	0	0bxx010000	0xYF F780	Channel A	CTS16A	CTS16A
						0bxx010001		Channel B	CTS16B	CTS16B
SASM	18	1	1	0	0	0bxx010010	0xYF F790	Channel A	CTS18A	CTS18A
						0bxx010011		Channel B	CTS18B	CTS18B
SASM	20	1	1	0	0	0bxx010100	0xYF F7A0	Channel A	CTS20A	CTS20A
						0bxx010101		Channel B	CTS20B	CTS20B

NOTES:
1. Interrupt arbitration priority goes in descending order with submodule #2 having the highest priority and the last interrupting submodule in the table having the lowest priority.
2. xx are the two VECT[7:6] bits contained in the BIUSM.
3. Y=m111, where m is the state of the modmap bit of the MCR of the SIM (Y=0x7 or 0xF).

9.1.2 CTM9 Pins and Naming Convention

The CTM9 uses 17 pins. The usage of these pins is shown in **Figure 9-1** and **Table 9-1**. The CTM9 digital input and output pin names are composed of three sections according to the following convention:

<submodule prefix><submodule number><submodule suffix (optional)>

The pin prefix and suffix for the different submodules used in the CTM9 are as follows:

- FCSM pin name:
  <submodule prefix>: “CTF”
  <submodule suffix>: none
  For example an FCSM placed as submodule number n would have its corresponding input clock pin called: CTFn

- MCSM pin name:
  <submodule prefix>: “CTM”
  <submodule suffix>: C for the Clock pin
  <submodule suffix>: L for the Load pin
  For example an MCSM placed as submodule number n would have its corresponding input clock pin named: CTMnC and its input load pin called CTMnL.
- SASM pin name:
  <submodule prefix>: “CTS”
  <submodule suffix>: A for channel A I/O
  <submodule suffix>: B for channel B I/O
  For example an SASM placed as submodule number n would have its corresponding channel A I/O pin named: CTSnA and its channel B I/O pin called: CTSnB.
- DASM pin name:
  <submodule prefix>: “CTD”
  <submodule suffix>: none
  For example a DASM placed as submodule number n would have its corresponding I/O pin called: CTDn
- PWMSM pin name:
  <submodule prefix>: “CPWM”
  <submodule suffix>: none
  For example a PWMSM placed as submodule number n would have its corresponding output pin called: CPWMn

In the CTM9, some pins are multiplexed between submodules. **Table 9-1** shows using the same pin names for the inputs or outputs which are connected together.

9.2 Free Running Counter Submodule (FCSM)

The free-running counter submodule (FCSM) has a 16-bit up counter with an associated clock source selector, selectable time-base bus drivers, software writable control registers, software readable status bits, and interrupt logic. When the 16-bit up counter overflows from 0xFFFF to 0x0000, an optional overflow interrupt is available to the software. The current state of the 16-bit counter is the primary output of the counter submodules. The software selects which, if any, time-base bus is to be driven by the 16-bit counter. A software control register selects whether the clock input to the counter is one of the taps from the prescaler or an input pin. The polarity of the external input pin is also programmable. The free-running counter submodule operation is comparable to the MC68HC11 counter.

A block diagram of the FCSM is shown in **Figure 9-2**. The main components of the FCSM are a 16-bit loadable free-running up-counter, a clock selector, a time base bus driver and an interrupt interface.

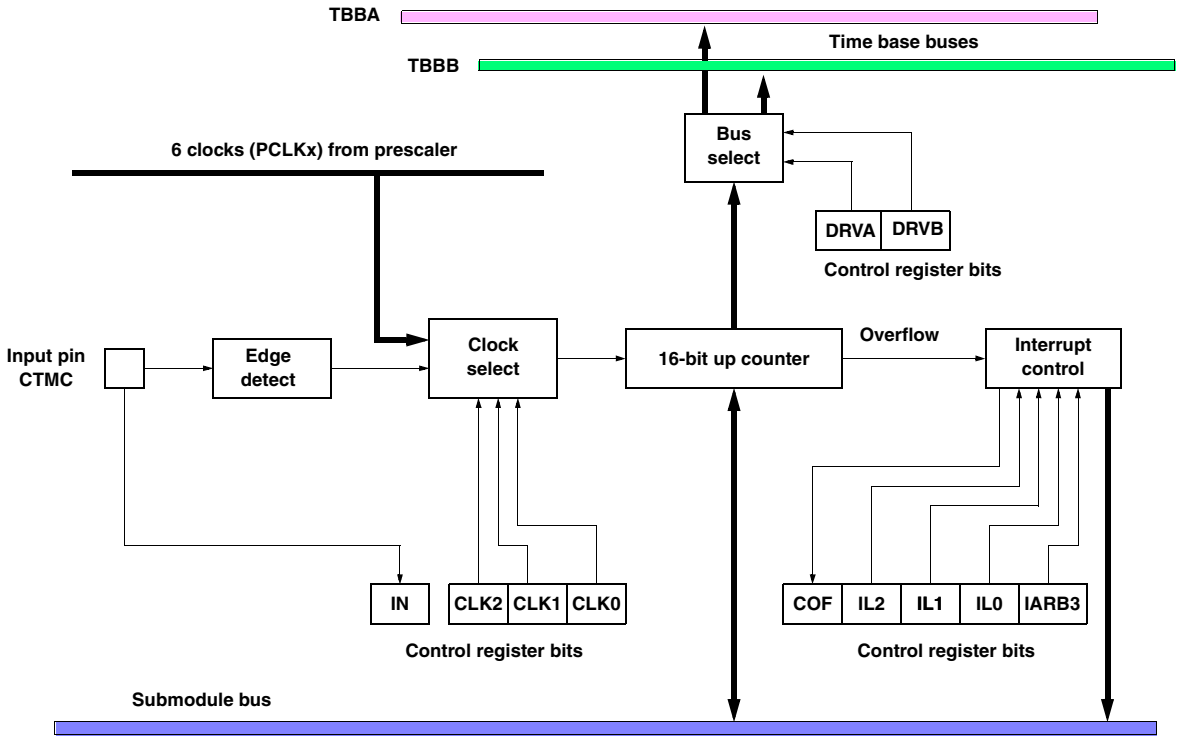


Figure 9-2 FCSM Block Diagram

NOTE

In order to be able to count, the FCSM requires the CPSM clock signals to be present. On coming out of reset, the FCSM will not count internal or external events until the prescaler in the CPSM starts running (when the software sets the PRUN bit). This allows all counters in the CTM submodules to be synchronized.

9.2.1 The FCSM Counter

The FCSM counter section comprises a 16-bit register and a 16-bit up-counter. Reading the register transfers the contents of the counter to the data bus, while a write to the register loads the counter with the new value. Overflow of the counter is defined to be the transition from 0xFFFF to 0x0000. An overflow condition causes the COF flag bit in the FCSMSIC register to be set.

NOTE

Reset presets the counter register to 0x0000. Writing 0x0000 to the counter register while the counter's value is 0xFFFF does not set the COF flag and does not generate an interrupt request.

9.2.2 FCSM Clock Sources

The user can choose from eight software selectable counter clock sources:

- Six prescaler outputs (PCLKx)
- Input pin rising edge detection on the input pin CTMC
- Input pin falling edge detection on the input pin CTMC

The clock source is selected by the CLK[2:0] bits in the FCSM status, interrupt and control register FCSMSIC (see **9.2.7.1 FCSMSIC — FCSM Status/Interrupt/Control Register**). When the CLK[2:0] bits are being changed, internal circuitry ensures that spurious edges occurring on the CTMC pin do not affect the FCSM.

NOTE

The read-only IN bit of the FCSMSIC register reflects the state of the input pin CTMC. The input pin is Schmitt triggered and is synchronized with the system clock (f_{SYS}).

9.2.3 FCSM External Event Counting

When an external clock source (on the input pin) is selected, the FCSM is in the event counter mode. The counter can simply count the number of events occurring on the input pin. Alternatively, the FCSM can be programmed to generate an interrupt when a predefined number of events have been counted; this is done by presetting the counter with the two's complement value of the desired number of events. When using the external clock source, the maximum guaranteed external frequency is $f_{SYS}/4$.

9.2.4 The FCSM Time Base Bus Driver

The DRVA and DRVB bits in the FCSMSIC register select the time base buses to be driven (see **9.2.7.1 FCSMSIC — FCSM Status/Interrupt/Control Register**).

WARNING

It is not recommended that the two time base buses be driven at the same time.

9.2.5 FCSM Interrupts

A valid FCSM interrupt can be generated when the COF bit in the FCSMSIC register is set (as a result of the counter overflowing). If the interrupt priority level of the FCSM is non-zero, as defined by the three IL bits in the FCSMSIC register, a valid interrupt request will occur on the IMB.

9.2.6 Freeze Action on the FCSM

When the IMB FREEZE signal is recognized, the FCSM counter stops counting and remains set at its current value. When the FREEZE signal is negated, the counter starts incrementing from its current value, as if nothing had happened. All registers are accessible during freeze.

During freeze, the IN bit in the FCSMSIC register continues to reflect the state of the signal on the input pin CTMC (see **9.2.7.1 FCSMSIC — FCSM Status/Interrupt/Control Register**).

9.2.7 FCSM Registers

The FCSM register map comprises four 16-bit register locations. As shown in **Table 9-2**, the register block contains two FCSM registers and two reserved registers. All unused bits and reserved address locations return zero when read by the software. Writing to unused bits and reserved address locations has no effect. In CTM implementations featuring multiple FCSMs, each FCSM has its own set of registers.

NOTE

All register addresses in this section are offsets from the base address of the FCSM.

Table 9-2 FCSM Register Map

Address	15 8	7 0
0xYF F760	Status, interrupt and control register (FCSMSIC)	
0xYF F762	Counter register(FCSMCNT)	

9.2.7.1 FCSMSIC — FCSM Status/Interrupt/Control Register

FMSMSIC — FCSM Status/Interrupt Control Register **0xYF F760**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
COF	IL2	IL1	IL0	IARB3	0	DRVA	DRVB	IN	0	0	0	0	CLK2	CLK1	CLK0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-3 FMSMSIC Bit Settings

Bit(s)	Name	Description
15	COF	Counter overflow flag. This status flag bit indicates whether or not a counter overflow has occurred. An overflow is defined to be the transition of the counter from 0xFFFF to 0x0000. If the IL field is non-zero, an interrupt request is generated when the COF bit is set. The flag clearing mechanism will work only if no flag setting event occurs between the read and write operations; if a COF setting event occurs between the read and write operations, the COF bit will not be cleared. 0 = Counter overflow has not occurred. 1 = Counter overflow has occurred.
14:12	IL[2:0]	Interrupt level. The three interrupt level bits are read/write control bits that select the priority level of interrupt requests made by the FCSM. These bits can be read or written at any time and are cleared by reset. 000 = Interrupt disabled 001 = Interrupt level 1 (lowest) 010 = Interrupt level 2 011 = Interrupt level 3 100 = Interrupt level 4 101 = Interrupt level 5 110 = Interrupt level 6 111 = Interrupt level 7 (highest)
11	IARB3	The read/write IARB3 bit works in conjunction with the IARB[2:0] field in the BIUSM module configuration register. Each module that generates interrupt requests on the IMB must have a unique value in the arbitration field (IARB). This interrupt arbitration identification number is used to arbitrate for the IMB when modules generate simultaneous interrupts of the same priority.
10	—	Reserved
9:8	DRV[A:B]	Drive time base bus. DRVA and DRVB are read/write bits that control the connection of the FCSM to the time base buses A and B. These bits are cleared by reset. It is recommended that the two time base buses **not** be driven at the same time. 00 = Neither time base bus A nor time base bus B is driven. 01 = Time base bus B is driven 10 = Time base bus A is driven 11 = Both time base bus A and time base bus B are driven
7	IN	Input pin status. This read-only status bit reflects the logic state of the FCSM input pin CTMC. Writing a ‘zero’ or a ‘one’ to this bit has no effect. Reset has no effect on this bit.
6:3	—	Reserved
2:0	CLK[2:0]	Counter clock select. These read/write control bits select one of six internal clock signals (PCLKx) or one of two external conditions on the input pin (rising edge or falling edge). The maximum frequency of the external clock signals is $f_{SYS}/4$. 000 = Prescaler output 1 (/2 or /3) 001 = Prescaler output 2 (/4 or /6) 010 = Prescaler output 3 (/8 or /12) 011 = Prescaler output 4 (/16 or /24) 100 = Prescaler output 5 (/32 or /48) 101 = Prescaler output 6 (/64 to /512 or /96 to /768) 110 = CTMC pin input, negative edge 111 = CTMC pin input, positive edge

9.2.7.2 FCSMCNT — FCSM Counter Register

FCSMCNT — FCSM Counter Register **0xYF F762**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MSB								LSB							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The FCSM counter register is a read/write register; it is cleared by reset.

9.3 Modulus Counter Submodule (MCSM)

The modulus counter submodule (MCSM) is an enhancement of the free-running counter. A modulus register gives the additional flexibility of recycling the counter at a count other than 64-Kbyte clock cycles. The state of the modulus register is transferred to the counter under three conditions:

1. When an overflow occurs.
2. When an appropriate transition occurs on the external load pin.
3. When the program writes to the counter register. In this case, the value is first written into the modulus register and immediately transferred to the counter.

Software can also write a value to the modulus register for later loading into the counter with one of the two first criteria. A block diagram of the MCSM is shown in **Figure 9-3**.

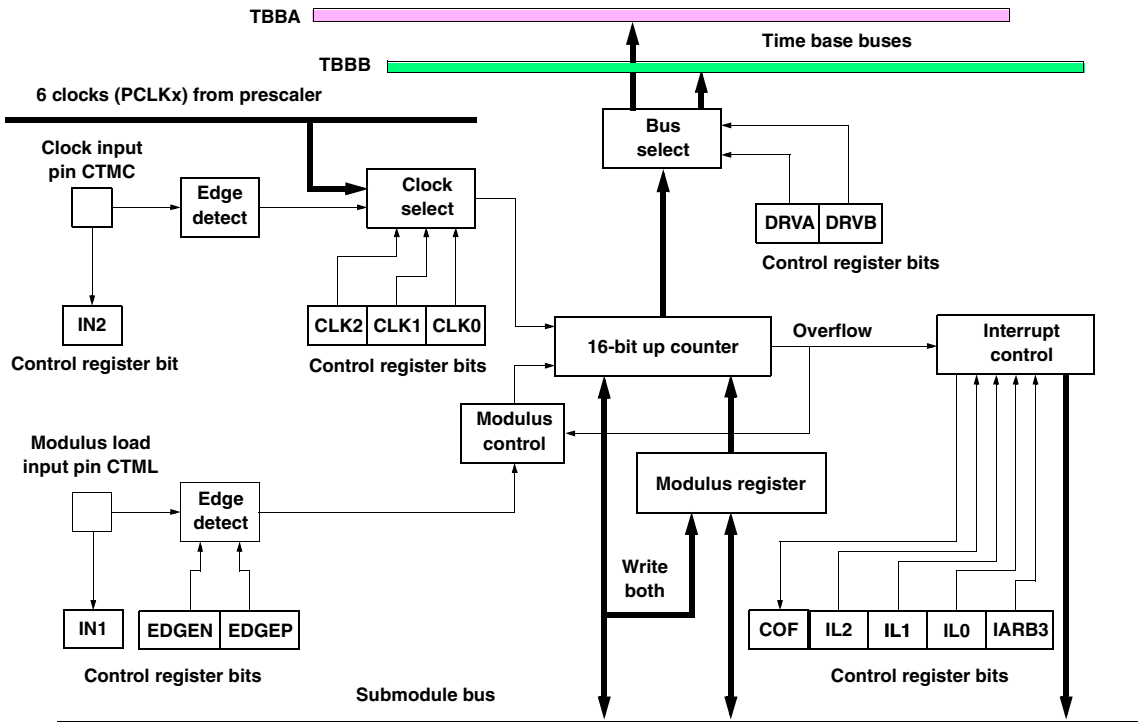


Figure 9-3 MCSM Block Diagram

The main components of the MCSM are a 16-bit modulus latch, a 16-bit loadable up-counter, counter loading logic, a clock selector, a time base bus driver and an interrupt interface.

NOTE

In order to be able to count, the MCSM requires the CPSM clock signals to be present. On coming out of reset, the MCSM will not count internal or external events until the prescaler in the CPSM starts running (when the software sets the PRUN bit). This allows all counters in the CTM submodules to be synchronized.

9.3.1 The MCSM Modulus Latch

The 16-bit modulus latch is a read/write register that is used to reload the counter automatically with a predetermined value. The contents of the modulus latch register can be read at any time. Writing to the register loads the modulus latch with the new value. This value is then transferred to the counter register on the next hardware load of that counter. However, writing to the corresponding counter register loads the modulus latch and the counter register immediately with the new value. The modulus latch register is cleared to 0x0000 by reset.

9.3.2 The MCSM Counter

The counter is composed of a 16-bit read/write register associated with a 16-bit incrementer. Reading the counter transfers the contents of the counter register to the data bus; writing to the counter loads the modulus latch and the counter register immediately with the new value. The counter can be clocked with different clock sources (see **9.3.3 MCSM Clock Sources**).

NOTE

Reset presets the counter register to 0x0000. Writing 0x0000 to the counter register while its value is 0xFFFF does not set the COF flag and does not generate an interrupt.

9.3.2.1 Loading the MCSM Counter Register

The counter register can be loaded by writing directly to it. The counter register is also loaded from the modulus latch each time a counter overflow occurs and the COF flag bit in the MCSM status/interrupt/control register (MCSMSIC) is set.

NOTE

When the modulus latch is loaded with 0xFFFF, the overflow flag is set on every counter clock pulse.

Loading of the counter register from the modulus register can also be triggered by an external event on the modulus load pin CTML. The edge on the CTML pin that triggers the loading of the counter register is selected by bits EDGEN and EDGEP in the MCSMSIC register. Hardware is provided to prevent the occurrence of spurious edges while changing the EDGEN and EDGEP bits. Reset clears the EDGEN and EDGEP

bits to zero, thereby preventing a signal on the CTML pin from loading the counter register until EDGEN and EDGEP have been initialized by the software. The modulus load input pin CTML is Schmitt triggered and synchronized to the system clock (f_{SYS}).

NOTE

The read-only IN1 bit of the MCSMSIC reflects the state of the input pin CTML.

9.3.2.2 Using the MCSM as a Free-Running Counter

The MCSM is a modulus counter. However it can be made to behave like a free-running counter by loading the modulus register with the value 0x0000.

9.3.3 MCSM Clock Sources

The User can choose from eight software selectable counter clock sources:

- Six prescaler outputs (PCLKx)
- Input pin rising edge detection on the input pin CTMC
- Input pin falling edge detection on the input pin CTMC

The clock source is selected by the CLK[2:0] bits in the MCSM status, interrupt and control register MCSMSIC (see **9.3.9 MCSMSIC — MCSM Status/Interrupt/Control Register**). When the CLK[2:0] bits are being changed, internal circuitry ensures that spurious edges occurring on the CTMC pin do not affect the MCSM. The clock input pin CTMC is Schmitt triggered and is synchronized with the system clock (f_{SYS}).

NOTE

The read-only IN2 bit of the MCSMSIC register reflects the state of the input pin CTMC.

9.3.4 MCSM External Event Counting

When an external clock source (on the CTMC input pin) is selected, the MCSM is in the event counter mode. The counter can simply count the number of events occurring on the input pin. Alternatively, the MCSM can be programmed to generate an interrupt when a predefined number of events have been counted; this is done by presetting the counter with the two's complement value of the desired number of events. When using the external clock source, the maximum external guaranteed frequency is $f_{SYS}/4$.

9.3.5 The MCSM Time Base Bus Driver

The DRVA and DRVB bits in the MCSMSIC register select the time base buses to be driven (see **9.3.9 MCSMSIC — MCSM Status/Interrupt/Control Register**).

9.3.6 MCSM interrupts

A valid MCSM interrupt can be generated when the COF bit in the MCSMSIC register is set as a result of the counter overflowing. If the interrupt priority level of the MCSM is non-zero, as defined by the three IL bits in the MCSMSIC register, a valid interrupt request will occur on the IMB.

9.3.7 Freeze Action on the MCSM

When the IMB FREEZE signal is recognized, the MCSM counter stops counting and remains set at its last value. When the FREEZE signal is negated, the counter starts incrementing from its last value, as if nothing had happened. All registers are accessible during freeze.

During freeze, the IN1 and IN2 bits in the MCSMSIC continue to reflect the states of the signals on the input pins (see **9.3.9 MCSMSIC — MCSM Status/Interrupt/Control Register**).

9.3.8 MCSM Registers

The MCSM register map comprises four 16-bit register locations. As shown in **Table 9-4**, the register block contains three FCSM registers and one reserved register. All unused bits and reserved address locations return zero when read by the software. Writing to unused bits and reserved address locations has no effect. In CTM implementations featuring multiple MCSMs, each MCSM has its own set of registers.

Table 9-4 MCSM Register Map

Address	15 8	7 0
0xYF F710	MCSM2 status/interrupt/control register (MCSM2SIC)	
0xYF F712	MCSM2 counter (MCSM2CNT)	
0xYF F714	MCSM2 modulus latch (MCSM2ML)	
0xYF F758	MCSM11 status/interrupt/control register (MCSM11SIC)	
0xYF F75A	MCSM11 counter (MCSM11CNT)	
0xYF F75C	MCSM11 modulus latch (MCSM11ML)	

9.3.9 MCSMSIC — MCSM Status/Interrupt/Control Register

MCSM2SIC — MCSM Status/Interrupt Control Register **0xYF F710**
MCSM11SIC **0xYF F758**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
COF	IL2	IL1	IL0	IARB3	0	DRVA	DRVB	IN2	IN1	EDGEN	EDGEP	0	CLK 2	CLK 1	CLK 0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-5 MCSMSIC Bit Settings

Bit(s)	Name	Description
15	COF	Counter overflow flag. This status flag bit indicates whether or not a counter overflow has occurred. An overflow of the MCSM counter is defined to be the transition of the counter from 0xFFFF to 0xxxxx, where 0xxxxx is the value contained in the modulus latch. If the IL field is non-zero, an interrupt request is generated when the COF bit is set. This flag bit is set only by the hardware and cleared only by the software or by a system reset. To clear the flag, the software must first read the bit (as '1') then write a '0' to the bit. The flag clearing mechanism will work only if no flag setting event occurs between the read and write operations; if a COF setting event occurs between the read and write operations, the COF bit will not be cleared. 0 = Counter overflow has not occurred 1 = Counter overflow has occurred.
14:12	IL[2:0]	Interrupt level. The three interrupt level bits are read/write control bits that select the priority level of interrupt requests made by the MCSM. These bits can be read or written at any time and are cleared by reset. 000 = Interrupt disabled. 001 = Interrupt level 1 (lowest). 010 = Interrupt level 2. 011 = Interrupt level 3. 100 = Interrupt level 4. 101 = Interrupt level 5. 110 = Interrupt level 6. 111 = Interrupt level 7 (highest).
11	IARB3	Interrupt arbitration bit 3. The read/write IARB3 bit works in conjunction with the IARB[2:0] field in the BIUSM module configuration register. Each module that generates interrupt requests on the IMB must have a unique value in the arbitration field (IARB). This interrupt arbitration identification number is used to arbitrate for the IMB when modules generate simultaneous interrupts of the same priority.
10	—	Reserved
9:8	DRV{A:B]	Drive time base bus. DRVA and DRVB are read/write bits that control the connection of the MCSM to the time base buses A and B. 00 = Neither time base bus A nor time base bus B is driven. 01 = Time base bus B is driven. 10 = Time base bus A is driven. 11 = Both time base bus A and time base bus B are driven.
7	IN2	Clock input pin status. This read-only status bit reflects the logic state of the clock input pin CTMC. Writing a 0 or 1 to this bit has no effect. Reset has no effect on this bit.
6	IN1	Modulus load input pin status. This read-only status bit reflects the logic state of the modulus load input pin CTML. Writing a 0 or 1 to this bit has no effect. Reset has no effect on this bit.
5:4	EDGEN, EDGEP	Modulus load edge sensitivity. These read/write bits select the sensitivity of the edge detection circuitry on the modulus load pin CTML. 00 = None 01 = Positive edge only. 10 = Negative edge only. 11 = Positive and negative edge.
3	—	Reserved
2:0	CLK[2:0]	Counter clock select. These read/write control bits select one of six internal clock signals (PCLKx) or one of two external conditions on the input pin (rising edges or falling edges). The maximum frequency of the external clock signals is $f_{SYS}/4$. 000 = Prescaler output 1 (/2 or /3). 001 = Prescaler output 2 (/4 or /6). 010 = Prescaler output 3 (/8 or /12). 011 = Prescaler output 4 (/16 or /24). 100 = Prescaler output 5 (/32 or /48). 101 = Prescaler output 6 (/64 to /768). 110 = CTMC pin input, negative edge. 111 = CTMC pin input, positive edge.

9.3.10 MCSMCNT — MCSM Counter Register

MCSM2CNT — MCSM Counter Register **0xYF F712**
MCSM11CNT **0xYF F75A**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MSB								LSB							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

9.3.11 MCSMML — MCSM Modulus Latch Register

MCSM2ML — MCSM Modulus Latch Register **0xYF F714**
MCSM11ML **0xYF F75C**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MSB								LSB							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

9.4 Single-Action Submodule (SASM)

The single action submodule provides an input capture and an output compare for each of two bidirectional pins. All of the functions associated with one pin comprise a SASM channel. Each channel includes a 16-bit equality comparator and one 16-bit register for saving an input capture value or for holding an output compare value. The input edge detector associated with each pin is programmable to cause the capture function to occur on the rising or falling edge. The output flip flop is set by the software to either toggle when an output compare occurs or to transfer a software provided bit value to the output pin. In either the input capture mode or the output compare mode, a software interrupt may be programmed to occur for each channel. Software selection is provided to select which of the two incoming time-base buses is used for input captures or output compares on each channel. Each channel operates independently. However, interrupt and interrupt priority logic are shared by both SASM channels. See **Figure 9-4** for a SASM block diagram.

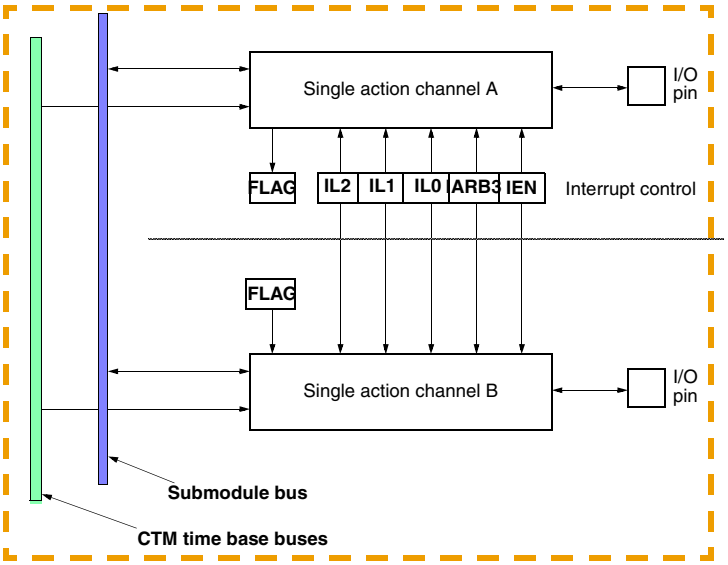


Figure 9-4 SASM Block Diagram

Each SASM channel comprises:

- A time base bus selector (which selects the time base bus to be used by that channel for all timing functions),
- A 16-bit data register (which can be read by the software at any time and which is used for both input capture and output compare functions),
- A 16-bit comparator (which continuously compares the 16-bit value in the data register with the time base bus),
- An output flip-flop (which holds the logic level to be sent to the output pin when a successful output compare occurs),
- An input edge detector (which detects the rising or falling edge that will trigger the input capture function),
- Several status and control bits in the status/interrupt/control register SICA or SICB,
- An interrupt section.

NOTE

During reset the output of the output flip-flop is cleared (i.e., to ‘0’).

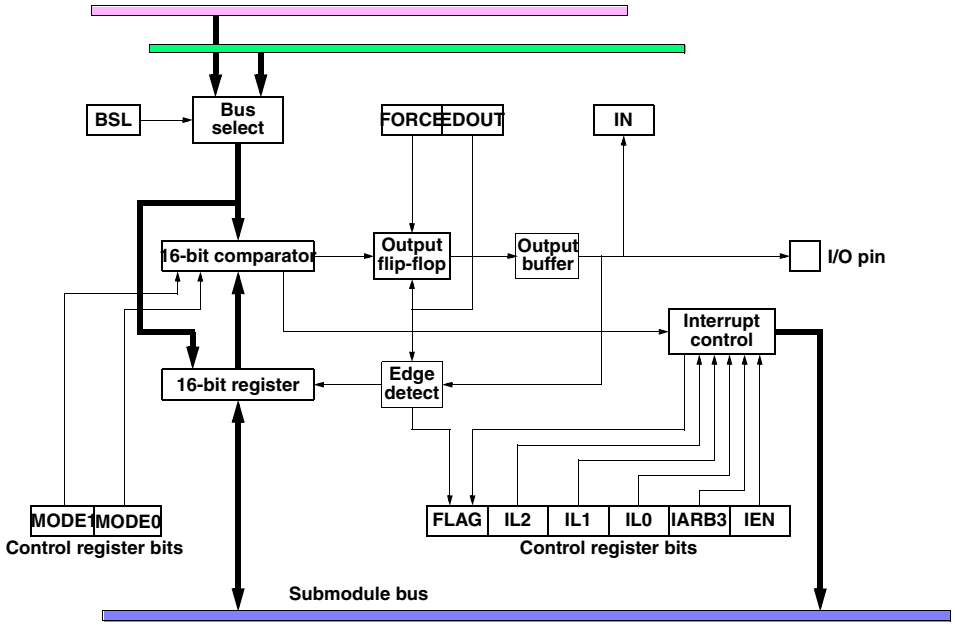


Figure 9-5 SASM Block Diagram (Channel A)

9.4.1 SASM Modes of Operation

Each SASM channel can operate in four different modes:

1. Input capture (IC) (i.e., either as input capture on a rising or falling edge or as a read-only input port)
2. Output compare (OC)
3. Output compare and toggle (OCT)
4. Output port (OP)

NOTE

For a channel operating in IC mode, the IN bit in the SIC register reflects the logic state of the corresponding input pin (after being Schmitt triggered and synchronized). When a channel is operating in OC, OCT or OP mode, the IN bit in the SIC register reflects the logic state of the output of the output flip-flop.

9.4.1.1 Clearing and Using the FLAG Bits

To clear a FLAG bit, the software must first read the channel's SIC register, then write a zero to the FLAG bit. These two steps do not have to be done on consecutive instructions. This clearing sequence must be used in every mode of operation. Writing a one to the FLAG bit has no effect.

WARNING

To avoid spurious interrupts, and to make sure that the FLAG bit is set according to the newly selected mode, the following sequence of operations should be adopted when changing mode:

1. Disable SASM interrupts
2. Change mode
3. Reset the corresponding FLAG bit
4. Re-enable SASM interrupts (if desired)

NOTE

When changing between output modes (OP, OC or OCT), it is not necessary to follow this procedure, as in these modes the FLAG bit merely indicates to the software that the compare value may be updated.

9.4.1.2 Input Capture (IC) Mode

In IC mode, the 16-bit counter value on the selected time base bus is 'captured' when a triggering event occurs on the channel's input pin. Triggering of the input capture circuitry is done by a rising or falling edge on the input pin; the polarity of the triggering edge is selected by the EDOUT bit. The logic level on the input pin can be read by software via the IN bit in the channel's SIC register.

In IC mode, the input pin is Schmitt triggered and the input signal is synchronized to the system clock (f_{SYS}). The IN bit reflects the state present on the input pin (after being Schmitt triggered and synchronized).

When an input capture occurs, the count value on the selected time base bus is latched into the channel's 16-bit data register. At the same time, the FLAG bit in the SIC register is set to indicate that an input capture has occurred.

The FLAG bit must be reset by software (see **9.4.1.1 Clearing and Using the FLAG Bits**). If the interrupt is serviced, the FLAG bit should be cleared by the servicing routine before returning from that routine. If a subsequent input capture event occurs while the FLAG bit is set, the new captured counter value is latched, and the FLAG bit remains unchanged.

In IC mode, the value of the EDOUT bit is permanently transferred to the output flip-flop. This value will be output on the pin when the mode is changed to one of the output modes.

9.4.1.3 Output Compare (OC) Mode

In OC mode, the state of an output pin is changed when a successful output compare occurs; an interrupt may also be generated. The output compare circuitry performs a comparison between the 16-bit register and the selected time base bus. When a match is found, the EDOUT bit value is transferred to the output flip-flop. At the same time, the FLAG bit is set to indicate to the processor that a match has occurred. Depending

on the state of the IEN bit, an interrupt can be generated when the FLAG bit is set. The FLAG bit must be reset by software (see **9.4.1.1 Clearing and Using the FLAG Bits**). If the interrupt is serviced, the FLAG bit should be cleared by the servicing routine before returning from that routine. If a subsequent output compare occurs while the FLAG bit is set, the output compare function occurs normally, and the FLAG bit remains set.

An output compare match can be simulated in software by writing a one to the FORCE bit. Setting the FORCE bit forces the EDOUT bit value onto the pin as if an output compare had occurred. In this case, the FLAG bit is not affected. Only if a genuine output compare occurs while doing a force, will the FLAG bit be set to signify that the compare has occurred.

In OC mode, the IN bit value reflects the logic state on the output of the output flip-flop.

9.4.1.4 Output Compare and Toggle (OCT) Mode

In OCT mode, the state of an output pin is toggled each time a successful output compare occurs; an interrupt may also be generated. The output compare circuitry performs a comparison between the 16-bit register and the selected time base bus. When a match is found, the output flip-flop is toggled to the opposite state. At the same time, the FLAG bit is set to indicate to the processor that the output compare has occurred. Depending on the state of the IEN bit, an interrupt can be generated when the FLAG bit is set. The FLAG bit must be reset by software (see **9.4.1.1 Clearing and Using the FLAG Bits**). If the interrupt is serviced, the FLAG bit should be cleared by the servicing routine before returning from that routine. If a subsequent output compare occurs while the FLAG bit is set, the output toggles, and the FLAG bit remains set.

An output compare match can be simulated in software by writing a one to the FORCE bit. Setting the FORCE bit forces the output flip flop to toggle as if an output compare had occurred. In this case, the FLAG bit is not affected. Only if a genuine output compare occurs while doing a force, will the FLAG bit be set to signify that the compare has occurred.

In OCT mode, the IN bit reflects the logic state on the output of the output flip-flop.

9.4.1.5 Output Port (OP) mode

In OP mode the channel's input/output pin is used as a single output port pin. The output compare function is still available, but for internal operation only, and does not affect the state of the output pin. An interrupt may also be generated when a compare occurs. The state of the output pin always reflects the value of the EDOUT bit in the channel's SIC register. Reading the EDOUT bit returns the last value written to it.

The internal compare feature compares the 16-bit register with the selected time base bus. The output compare circuitry performs a comparison between the 16-bit register and the selected time base bus. When a match is found, the FLAG bit is set to indicate to the processor that the output compare has occurred. Depending on the state of the IEN bit, an interrupt can be generated when the FLAG bit is set. The FLAG bit must be reset by software (see **9.4.1.1 Clearing and Using the FLAG Bits**). If the interrupt is

serviced, the FLAG bit should be cleared by the servicing routine before returning from that routine. If a subsequent output compare occurs while the FLAG bit is set, the internal output compare functions normally, and the FLAG bit remains set.

In OP mode, the IN bit value reflects the logic state on the output of the output flip-flop.

9.4.2 SASM Interrupts

Each channel in the dual-channel SASM has separately enabled and initiated interrupts and they each have their own unique vector number and address. However, they are both assigned to the same interrupt level and arbitration priority by the IL[2:0] and IARB3 bits in the SICA register.

A valid SASM interrupt is recognized when the FLAG bit is set, the corresponding IEN bit is set and the interrupt level defined by bits IL[2:0] is not equal to zero.

The FLAG bit is a status bit that indicates, when set, that an input capture or output compare has occurred on the corresponding single action channel.

The relative priority of these sources of interrupt is fixed and channel A has a higher priority than channel B.

9.4.3 Freeze Action on the SASM

When the IMB FREEZE signal is recognized, the SASM input capture and output compare functions are halted. As soon as the FREEZE signal is negated, SASM actions resume as if nothing had happened. During freeze, the IN bits of the SIC registers (SICA and SICB) are readable and return the levels present at the input pins if an input mode is in operation, or the output value if an output mode is in operation (see **9.4.4.1 SICA — SASM Status/Interrupt Control Register A** and **9.4.4.3 SICB — SASM Status/Interrupt Control Register B**). When one of the output modes is in operation, the force output function remains available, allowing the software to output the desired level (a useful feature for debugging). All SASM registers are accessible during freeze.

9.4.4 SASM Registers

The SASM register map comprises eight 16-bit register locations. As shown in **Table 9-6**, the register block contains two SASM registers for each channel and four reserved registers. All unused bits and reserved address locations return zero when read by the software. Writing to unused bits and reserved address locations has no meaning nor effect. All register addresses in this section are specified as offsets from the base address of the SASM. In CTM implementations featuring multiple SASMs, each SASM has its own set of registers.

Table 9-6 SASM Register Map

Address	15 8	7 0
0xYF F770	SASM14 status/interrupt/control register A (S14ICA)	
0xYF F772	SASM14 data register A (S14DATA)	
0xYF F774	SASM14 status/interrupt/control register B (S14ICB)	
0xYF F776	SASM14 data register B (S14DATB)	
0xYF F780	SASM16 status/interrupt/control register A (S16ICA)	
0xYF F782	SASM16 data register A (S16DATA)	
0xYF F784,	SASM16 status/interrupt/control register B (S16ICB)	
0xYF F786	SASM16 data register B (S16DATB)	
0xYF F790	SASM18 status/interrupt/control register A (S18ICA)	
0xYF F792	SASM18 data register A (S18DATA)	
0xYF F794	SASM18 status/interrupt/control register B (S18ICB)	
0xYF F796	SASM18 data register B (S18DATB)	
0xYF F7A0	SASM20 status/interrupt/control register A (S20ICA)	
0xYF F7A2	SASM20 data register A (S20DATA)	
0xYF F7A4	SASM20 status/interrupt/control register B (S20ICB)	
0xYF F7A6	SASM20 data register B (S20DATB)	

9.4.4.1 SICA — SASM Status/Interrupt Control Register A

This register contains the control, interrupt enable and status bits for SASM channel A. It also contains the interrupt priority level bits IL[2:0] and the arbitration priority bit IARB3 for the whole SASM (i.e., common to channels A and B).

S14ICA — SASM Status/Interrupt Control Register A **0xYF F770**
S16ICA **0xYF F780**
S18ICA **0xYF F790**
S20ICA **0xYF F7A0**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
FLAG	IL2	IL1	IL0	IARB3	IEN	0	BSL	IN	0	FORCE	EDOUT	0	0	MODE1	MODE0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-7 SICA Bit Settings

Bit(s)	Name	Description
15	FLAG	Event flag. The FLAG bit is set whenever an input capture or output compare event occurs. This flag bit is set only by the hardware and cleared only by the software or by a system reset. If the IL field is non-zero, and the IEN bit is set, an interrupt request is generated when the FLAG bit is set. In IC mode, if a subsequent input capture event occurs while the FLAG bit is set, the new value is latched and the FLAG bit remains set. In OC mode, if a subsequent output compare event occurs while the FLAG bit is set, the compare occurs normally and the FLAG bit remains set. In OCT mode, if a subsequent output compare event occurs while the FLAG bit is set, the toggle of the output signal occurs as normal and the FLAG bit remains set. In OP mode, if a subsequent internal compare event occurs while the FLAG bit is set, the compare occurs normally and the FLAG bit remains set. To clear the flag, the software must first read the bit (as '1') then write a '0' to the bit. The flag clearing mechanism will work only if no flag setting event occurs between the read and write operations; if a FLAG setting event occurs between the read and write operations, the FLAG bit will not be cleared. 0 = An input capture or output compare event has not occurred. 1 = An input capture or output compare event has occurred.
14:12	IL[2:0]	Interrupt level. The three interrupt level bits are read/write control bits that select the priority level of interrupt requests made by the SASM. These bits can be read or written at any time and are cleared by reset. These bits affect both SASM channels, not just channel A. 000 = Interrupt disabled. 001 = Interrupt level 1 (lowest). 010 = Interrupt level 2. 011 = Interrupt level 3. 100 = Interrupt level 4. 101 = Interrupt level 5. 110 = Interrupt level 6. 111 = Interrupt level 7 (highest).
11	IARB3	Interrupt arbitration bit 3. The read/write IARB3 bit works in conjunction with the IARB[2:0] field in the BIUSM module configuration register. Each module that generates interrupt requests on the IMB must have a unique value in the arbitration field (IARB). This interrupt arbitration identification number is used to arbitrate for the IMB when modules generate simultaneous interrupts of the same priority. This bit affects both SASM channels, not just channel A.
10	IEN	Interrupt enable. This control bit enables interrupts on channel A when the FLAG bit is set and the IL[2:0] field is non-zero. This bit is cleared by reset 0 = Interrupts disabled. 1 = Interrupts enabled.
9	—	Reserved
8	BSL	Time base bus select. This control bit selects the time base bus to be connected to SASM channel A. This bit is cleared by reset. 0 = Time base bus A selected. 1 = Time base bus B selected.
7	IN	Input pin status. In input mode (IC), the IN bit reflects the logic state present on the corresponding input pin (after being Schmitt triggered and synchronized). In the output modes (OC, OCT and OP), the IN bit value reflects the state of the output of the output flip-flop. The IN bit is a read-only bit; writing to it has no effect. Reset has no effect on this bit.
6	—	Reserved
5	FORCE	Supervisor/user data space. The SUPV bit places the SCIM2E global registers in either supervisor or user data space. The FLAG bit is not affected by the use of the FORCE bit. 0 = No action 1 = Force output flip-flop to behave as if an output compare has just occurred.

Table 9-7 SICA Bit Settings (Continued)

Bit(s)	Name	Description
4	EDOUT	Edge detect and output level. In IC mode, the EDOUT bit is used to select the edge that will trigger the input capture circuitry. In OC mode, the EDOUT bit is used to latch the value to be output to the pin on the next output compare match or when the FORCE bit is set. Internal synchronization ensures that the correct level appears on the output pin when a new value is written to EDOUT and FORCE is set at the same time. Reading EDOUT returns the previous value written. In OCT mode, the EDOUT bit has no effect. However, the force function is still available and will force the value of the EDOUT bit to appear on the output pin. In OP mode, the value of the EDOUT bit is output to the corresponding pin. Reading EDOUT returns the previous value written. 0 = Input capture on falling edge. 1 = Input capture on rising edge.
3:2	—	Reserved
1:0	MODE1, MODE0	SASM operating mode select. These control bits select the mode of operation of the SASM channel, as shown in the following table. MODE1 and MODE0 are cleared by reset. 00 = Input capture (IC). 01 = Output port (OP). 10 = Output compare (OC 11 = Output compare and toggle (OCT)

9.4.4.2 SDATA — SASM Data Register A

SDATA is the 16-bit read-write register associated with channel A. In IC mode, SDATA contains the last captured value. In the OC, OCT and OP modes, it is loaded with the value of the next output compare. SDATA is not affected by reset.

S14DATA — SASM Data Register A **0xYF F772,**
S16DATA **0xYF F782**
S18DATA **0xYF F792**
S20DATA **0xYF F7A2**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MSB								LSB							
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

9.4.4.3 SICB — SASM Status/Interrupt Control Register B

This register contains the control and status bits for SASM channel B. The bits it contains are identical to those in SICA, with the exception of the IL[2:0], IARB3 and IEN which apply to both channels simultaneously and which are included only in SICA. For descriptions of the bits, please refer to **9.4.4.1 SICA — SASM Status/Interrupt Control Register A**).

S14ICB — SASM Status/Interrupt Control Register B **0xYF F774**
S16ICB **0xYF F784**
S18ICB **0xYF F794**
S20ICB **0xYF F7A4**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
FLAG	0	0	0	0	0	0	BSL	IN	0	FORCE	EDOUT	0	0	MODE1	MODE0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

9.4.4.4 SDATB — SASM Data Register B

SDATB is the 16-bit read-write register associated with channel A. In the IC mode, SDATB contains the last captured value. In the OC, OCT and OP modes, it is loaded with the value of the next output compare. SDATB is not affected by reset.

S14DATB — SASM Data Register B **0xYF F776**
S16DATB **0xYF F786**
S18DATB **0xYF F796**
S20DATB **0xYF F7A6**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MSB								LSB							
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

9.5 Double-Action Submodule (DASM)

The double-action submodule (DASM) provides two 16-bit input captures or two 16-bit output compare functions that can occur automatically without software intervention. The input edge detector is programmable to cause the capture function to occur on the desired edges. The output flip-flop is set by one of the output compares and is reset by the other one. In either the input capture modes or the output compare modes, an optional interrupt is available to the software. Software selection is provided for which of two incoming time-base buses is used for input captures or output compares.

The DASM can work in six different modes: disable mode, pulse length measurement, period measurement, input capture mode, single pulse generation, and continuous pulse width generation.

The DASM has three data registers that are accessible to the software from the various modes. For some of the modes, two of the registers are cascaded together to provide double buffering. The value in one register is transferred to another register automatically at the correct time so that the minimum pulse (measurement or generation) is just one time-base bus count. See **Figure 9-6** for a DASM block diagram.

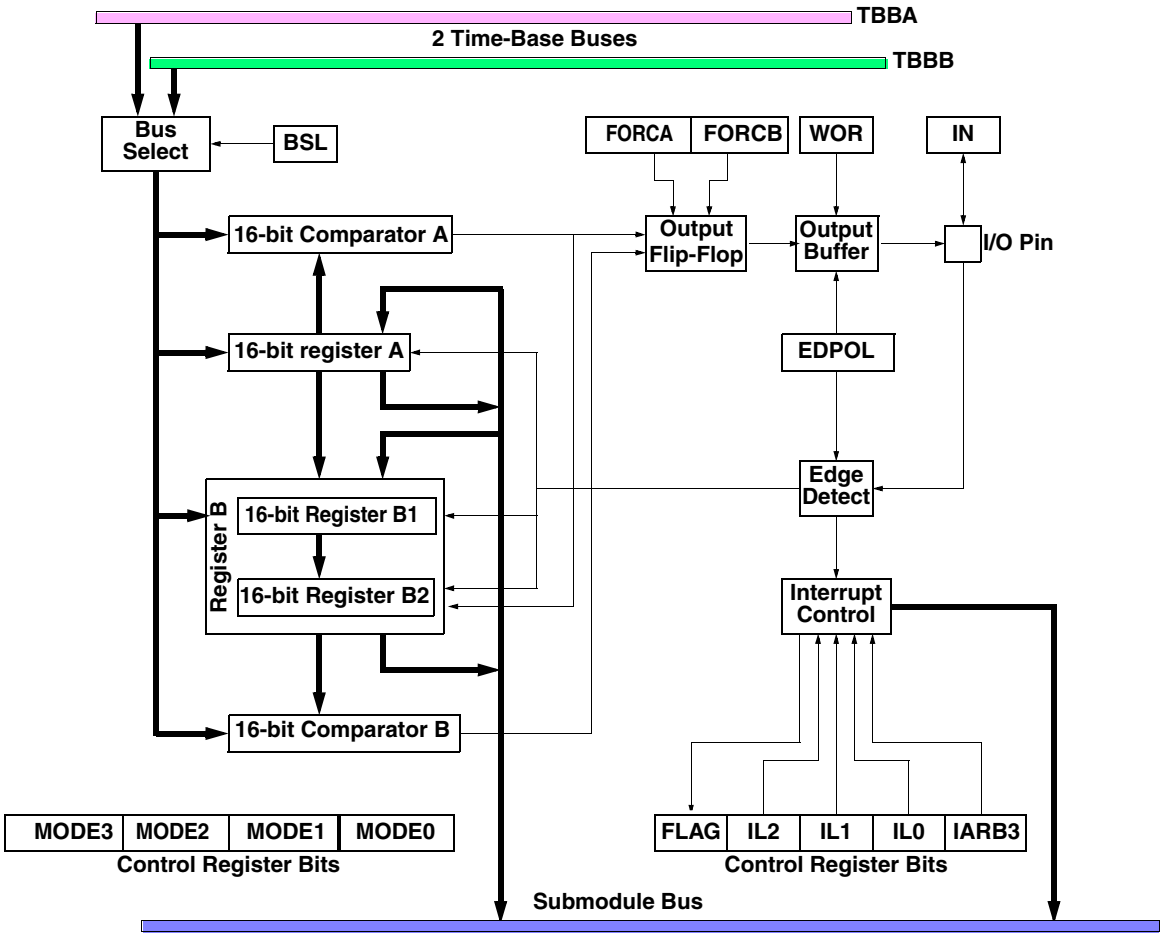


Figure 9-6 DASM Block Diagram

Channel A comprises one 16-bit data register and one 16-bit comparator. Channel B also appears to the user to consist of one 16-bit data register and one 16-bit comparator, however, internally, channel B has two data registers B1 and B2, and the operating mode determines which register is accessed by the software:

- In the input capture modes (IPWM, IPM and IC), registers A and B2 are used to hold the captured values; in these modes, the B1 register is used as a temporary latch for channel B.
- In the output compare modes (OCA and OCAB), registers A and B2 are used to define the output pulse; register B1 is not used in these modes.
- In the output pulse width modulation mode (OPWM), registers A and B1 are used as primary registers and hidden register B2 is used as a double buffer for channel B.

Register contents are always transferred automatically at the correct time so that the minimum pulse (measurement or generation) is just one time base bus count. The A and B data registers are always read/write registers, accessible via the CTM's submodule bus.

In the input capture modes, the edge detect circuitry triggers a capture whenever a rising or falling edge (as defined by the EDPOL bit) is applied to the input pin. The signal on the input pin is Schmitt triggered and synchronized with the system clock (f_{SYS}).

In the disabled mode (DIS) and in the input modes, the IN bit reflects the state present on the input pin (after being Schmitt triggered and synchronized). In the output modes the IN bit reflects the value present at the output of the output flip-flop. The output flip-flop is used in output modes to hold the logic level applied to the output pin.

The time base bus selector is common to all input and output functions; it connects the DASM to time base bus A or B and is controlled in software by the bus select bit BSL in the DASMSIC register.

9.5.1 32-Bit Coherent Access

In the IPWM and IPM modes, 32-bit coherent access of the data registers is supported. A 32-bit coherent access consists of doing a long word aligned access of data register A. In this case, register A is accessed first, immediately followed (on the next cycle) by a register B access. During this time, any flag setting or data transfer from the hidden B register is deferred until coherent access has ended. When the 32-bit access has ended, the DASM finishes any pending B action and resumes normal operation.

9.5.2 DASM Modes of Operation

The mode of operation of the DASM is determined by the mode select bits MODE[3:0] in the DASMSIC register (see **Table 9-8**).

Table 9-8 DASM Modes of Operation

MODE[3:0]	Mode	Description of mode
0000	DIS	Disabled — Input pin is high impedance; IN gives state of the input pin.
0001	IPWM	Input pulse width measurement — Capture on the leading edge and the trailing edge of an input pulse.
0010	IPM	Input period measurement — Capture two consecutive rising/falling edges.
0011	IC	Input capture — Capture when the designated edge is detected.
0100	OCB	Output compare, flag set on B compare — Generate leading and trailing edges of an output pulse and set the flag.
0101	OCAB	Output compare, flag on A and B compare — Generate leading and trailing edges of an output pulse and set the flag.
1xxx	OPWM	Output pulse width modulation — Generate continuous PWM output with 7, 9, 11, 12, 13, 14, 15 or 16 bits of resolution.

WARNING

To avoid spurious interrupts, and to make sure that the FLAG bit is set according to the newly selected mode, the following sequence of operations should be adopted when changing mode:

1. Disable DASM interrupts
2. Change mode
3. Reset the corresponding FLAG bit
4. Re-enable DASM interrupts (if desired)

NOTE

When changing between output modes (OP, OC or OCT), it is not necessary to follow this procedure, as in these modes the FLAG bit merely indicates to the software that the compare value can be updated.

9.5.2.1 Disable (DIS) mode

DIS mode is selected by making MODE[3:0] = 0000.

In this mode, all input capture and output compare functions of the DASM are disabled and the FLAG bit is maintained in its reset state, but the input port pin function remains available. The associated pin becomes a high impedance input and the input level on this pin is reflected by the state of the IN bit in the DASMSIC register. All control and interrupt bits remain accessible, allowing the software to prepare for future mode selection. Data registers A and B are accessible at consecutive addresses. Writing to data register B stores the same value in registers B1 and B2.

WARNING

When changing modes, it is imperative to go through the DIS mode in order to reset the DASM's internal functions properly. Failure to do this could lead to invalid and unexpected output compare or input capture results, and to flags being set incorrectly.

9.5.2.2 Input Pulse Width Measurement (IPWM) Mode

IPWM mode is selected by making MODE[3:0] = 0001.

This mode allows the width of a positive or negative pulse to be determined by capturing the leading edge of the pulse on channel B and the trailing edge of the pulse on channel A; successive captures are done on consecutive edges of opposite polarity. The edge sensitivity is selected by the EDPOL bit in the DASMSIC register.

This mode also allows the software to determine the logic level on the input pin at any time by reading the IN bit in the DASMSIC register.

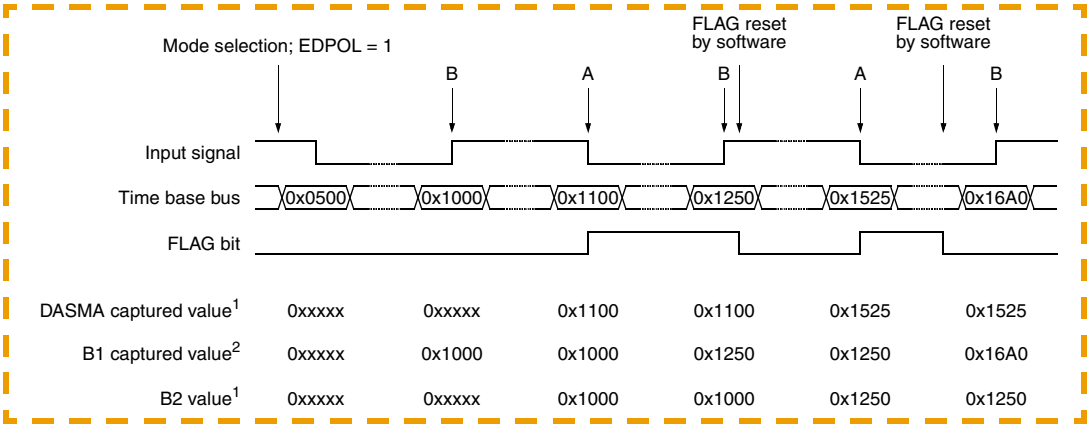
The channel A input capture function remains disabled until the first rising edge triggers the first input capture on channel B. When this rising edge is detected, the count value of the time base bus selected by the BSL bit is latched in the 16-bit data register

B1; the FLAG bit is not affected. When the next falling edge is detected, the count value of the time base bus is latched into the 16-bit data register A and, at the same time, the FLAG bit is set and the contents of register B1 are transferred to register B2. Reading data register B returns the value in register B2. If subsequent input capture events occur while the FLAG bit is set, data registers A and B will be updated with the latest captured values and the FLAG bit will remain set.

If a 32-bit coherent operation is in progress when the falling edge is detected, the transfer from B1 to B2 is deferred until the coherent operation is completed. Operation of the DASM then continues on channels B and A as previously described.

The input pulse width is calculated by subtracting the value in data register B from the value in data register A.

Figure 9-7 provides an example of how the DASM can be used for input pulse width measurement.



Notes: 1. These values are accessible to the software.
2. These values are internal and are not accessible.

Figure 9-7 Input Pulse Width Measurement Example

9.5.2.3 Input Period Measurement (IPM) Mode

IPM mode is selected by making MODE[3:0] = 0010.

This mode allows the period of an input signal to be determined by capturing two consecutive rising edges or two consecutive falling edges; successive input captures are done on consecutive edges of the same polarity. The edge polarity is defined by the EDPOL bit in the DASMSIC register.

This mode also allows the software to determine the logic level on the input pin at any time by reading the IN bit in the DASMSIC register.

When the first edge having the selected polarity is detected, the time base bus value is latched into the 16-bit data register A, the data in register B1 is transferred to data register B2 and finally the data in register A is transferred to register B1. On this first

capture the FLAG bit is not set. On the second and subsequent captures, the FLAG bit is set immediately before the data in register A is transferred to register B1.

When the second edge of the same polarity is detected, the time base bus value is latched into data register A, the data in register B1 is transferred to data register B2, the FLAG bit is set to signify that the beginning and end points of a complete period have been captured, and finally data register A is transferred to register B1. This sequence of events is repeated for each subsequent capture. Reading data register B returns the value in register B2.

If a 32-bit coherent operation is in progress when an edge is detected, the transfer of data from B1 to B2 is deferred until the coherent operation is completed. At any time, the input level present on the input pin can be read on the IN bit.

The input pulse period is calculated by subtracting the value in data register B from the value in data register A.

Figure 9-8 provides an example of how the DASM can be used for input period measurement.

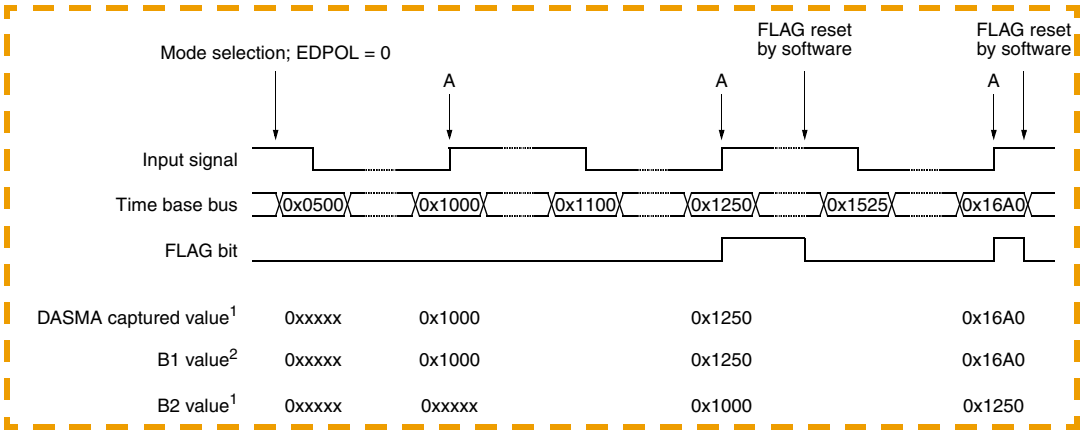


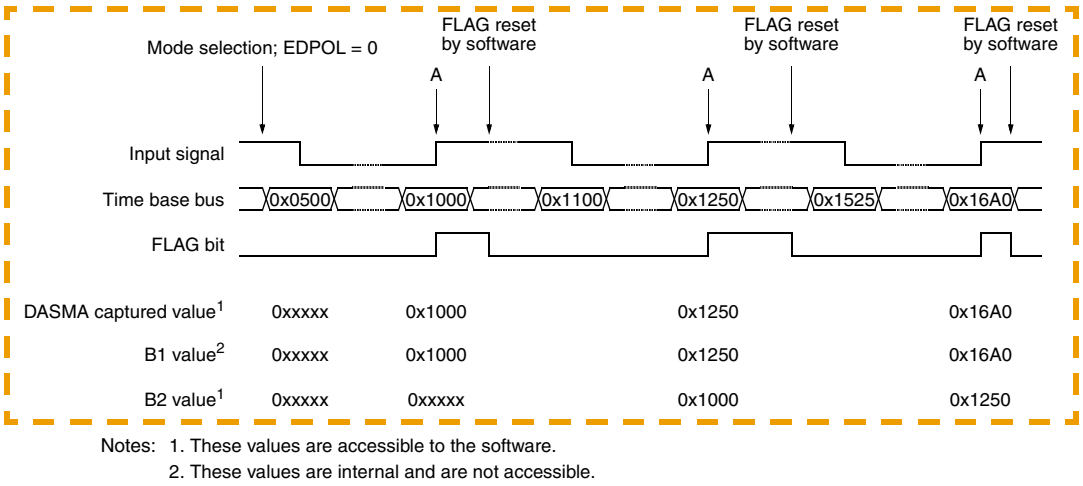
Figure 9-8 Input Period Measurement Example

9.5.2.4 Input Capture (IC) Mode

IC mode is selected by making MODE[3:0] = 0011.

This mode is identical to the input period measurement mode (IPM) described above, with the exception that the FLAG bit is also set at the occurrence of the first detected edge of the selected polarity. In this mode the DASM functions as a standard input capture function in a similar way to the M68HC11 family timers. In this case the value latched in channel B can be ignored.

Figure 9-9 provides an example of how the DASM can be used for input capture.



Notes: 1. These values are accessible to the software.
2. These values are internal and are not accessible.

Figure 9-9 DASM Input Capture Example

9.5.2.5 Output Compare (OCB and OCAB) Modes

OC mode is selected by making MODE[3:0] = 010x. The MODE0 bit controls the setting criteria for the FLAG bit, (i.e., when a compare occurs only on channel B or when a compare occurs on either channel). See **9.5.5.1 DASMSIC — DASM Status/Interrupt Control Register**.

This mode allows the DASM to perform four different output functions:

- Single-shot output pulse (two edges), with FLAG set on the second edge.
- Single-shot output pulse (two edges), with FLAG set on both edges.
- Single-shot output transition (one edge).
- Output port pin, with output compare function disabled.

In this mode the leading and trailing edges of variable width output pulses are generated by calculated output compare events occurring on channels A and B, respectively. OC mode may also be used to perform a single output compare function, similar to the M68HC11 timer, or may be used as an output port bit.

In this mode, channel B is accessed via register B2. Register B1 is not used and is not accessible to the user. Both channels work together to generate one ‘single shot’ output pulse signal. Channel A defines the leading edge of the output pulse, while channel B defines the trailing edge of the pulse. FLAG setting can be done when a compare occurs on channel B only or when a compare occurs on either channel (as defined by the MODE0 bit in the DASMSIC register).

When this mode is first selected, both comparators are disabled. Each comparator is enabled by writing to its data register; it remains enabled until the next successful comparison is made on that channel, whereupon it is disabled. The values stored in registers A and B are compared with the count value on the selected time base bus when their corresponding comparators are enabled.

The output flip-flop is set when a match occurs on channel A. The output flip-flop is reset when a match occurs on channel B. The polarity of the output signal is selected by the EDPOL bit. The output flip-flop level can be obtained at any time by reading the IN bit.

If subsequent enabled output compares occur on channels A and B, the output pulses continue to be output, regardless of the state of the FLAG bit.

At any time, the FORCA and FORCB bits allow the software to force the output flip-flop to the level corresponding to a comparison on channel A or B, respectively. Note that the FLAG bit is not affected by these 'force' operations.

Totem pole or open-drain output circuit configurations can be selected using the WOR bit in the DASMSIC register.

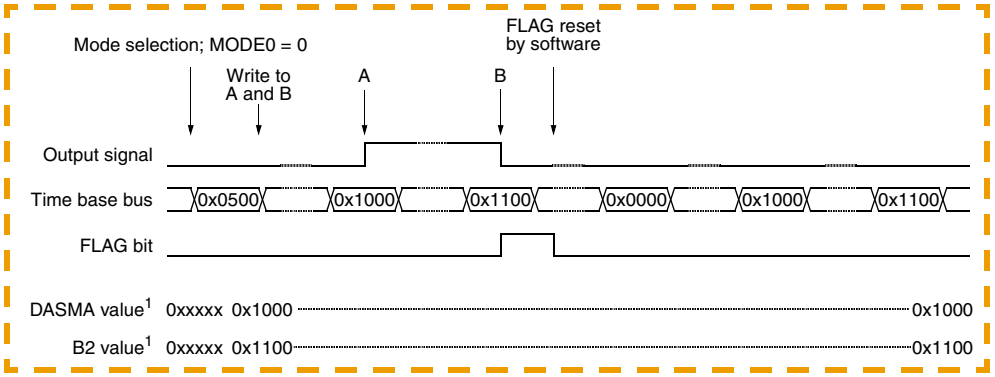
WARNING

There is no hardware protection to disable comparator B while comparator A is enabled. It is the user's responsibility to load data registers A and B with the values needed to produce the desired output pulse.

NOTE

If both channels are loaded with the same value they will try to force different levels on the output flip-flop. Hardware protection circuitry ensures that no contention occurs and the output flip-flop provides a logic zero level output.

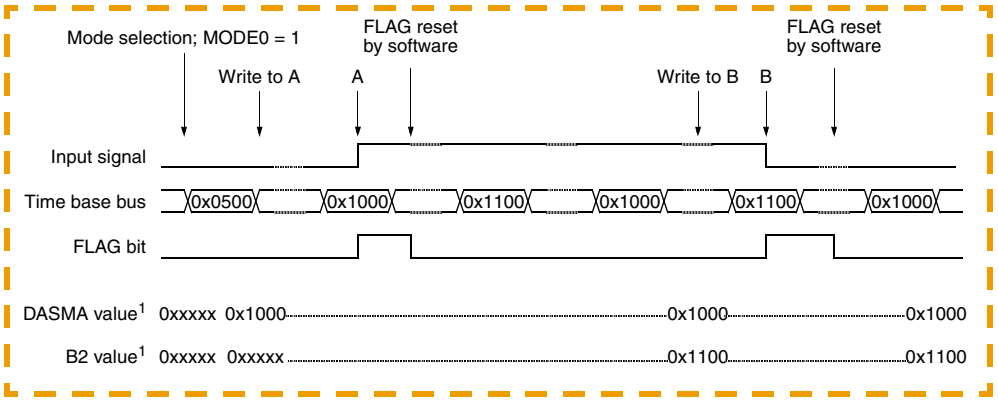
Single Shot Output Pulse Operation — The single shot output pulse operation is selected by writing the leading edge value of the desired pulse to data register A and the trailing edge value to data register B. A single pulse will be output at the desired time, thereby disabling the comparators until new values are written to the data registers. In this mode, registers A and B2 are accessible to the user software (at consecutive addresses). **Figure 9-10** provides an example of how the DASM can be used to generate a single output pulse.



Note: 1. These values are accessible to the software.

Figure 9-10 Single Shot Output Pulse Example

Single Output Compare Operation — The single output compare operation is selected by writing to only one of the two data registers (A or B), thus enabling only one of the comparators. Following the first successful match on the enabled channel, the output level is fixed and remains at the same level indefinitely with no further software intervention being required. In this mode, registers A and B2 are accessible to the user software (at consecutive addresses). Figure 9-11 provides an example of how the DASM can be used to perform a single output compare.



Note: 1. These values are accessible to the software.

Figure 9-11 Single Shot Output Transition Example

Output Port Bit Operation — The output port bit operation is selected by leaving both channels disabled, (i.e., by writing to neither register A nor B). The EDPOL bit alone controls the output value. The same result can be achieved by keeping EDPOL at zero and using the FORCA and FORCB bits to obtain the desired output level.

9.5.2.6 Output Pulse Width Modulation (OPWM) Mode

OPWM mode is selected by making MODE[3:0] = 1xxx. The MODE[2:0] bits allow some of the comparator bits to be masked.

This mode allows pulse width modulated output waveforms to be generated, with eight selectable frequencies (for a given time base). Both channels (A and B) are used to generate one PWM output signal on the DASM pin.

Channel B is accessed via register B1. Register B2 is not accessible to the user. Channels A and B define the leading and trailing edges, respectively, of the PWM output pulse. The value in register B1 is continuously transferred to register B2 in the time between each trailing edge and the following leading edge.

The value loaded in register A is continuously compared with the value on the time base bus. When a match on A occurs, the FLAG bit is set and the output flip-flop is set. The value loaded in register B2 is continually compared with the value on the time base bus. When a match occurs on B, the output flip-flop is reset.

The polarity of the PWM output signal is selected by the EDPOL bit. The output flip-flop level can be obtained at any time by reading the IN bit.

If subsequent compares occur on channels A and B, the PWM pulses continue to be output, regardless of the state of the FLAG bit.

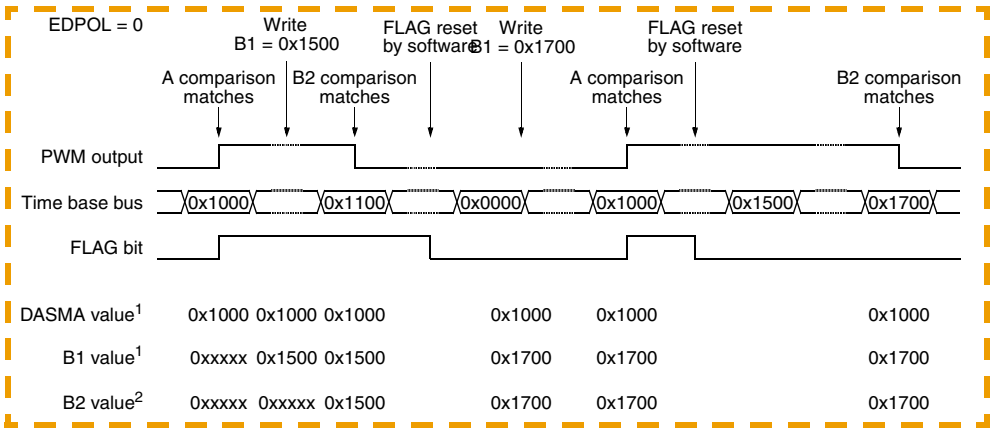
At any time, the FORCA and FORCB bits allow the software to force the output flip-flop to the level corresponding to comparison on A or B respectively. Note that the FLAG bit is not affected by the FORCA and FORCB operations.

WARNING

There is no hardware protection to disable comparator B while comparator A is enabled. It is the user's responsibility to load data registers A and B with the values needed to produce the desired PWM output pulse.

If both channels are loaded with the same value they will try to force different levels on the output flip-flop. Hardware protection circuitry ensures that no contention occurs and the output flip-flop provides a logic zero level output.

Figure 9-12 provides an example of how the DASM can be used for pulse width modulation.



Notes: 1. These values are accessible to the software.
2. These values are internal and are not accessible.

Figure 9-12 DASM Output Pulse Width Modulation Example

To generate PWM output pulses of different frequencies, the 16-bit comparator can have some of its bits masked. This is controlled by bits MODE2, MODE1 and MODE0. The frequency of the PWM output (f_{PWM}) is given by equation 1 (assuming the DASM is connected to a free running counter):

$$f_{PWM} = \frac{f_{SYS}}{N_{CPSM} \bullet N_{DASM}}$$

where N_{CPSM} is the overall CPSM clock divide ratio (÷2 to ÷512 or ÷3 to ÷768) and N_{DASM} is the DASM divide ratio.

A few examples of frequencies and resolutions that can be obtained are shown in **Table 9-9**.

Table 9-9 DASM PWM Example Output Frequencies/Resolutions at f_{SYS} = 16 MHz

N_{CPSM}	N_{DASM}[1]	PWM output frequency (Hz)	Resolution (bits)
512	65536	0.48	16
2	65536	122.07	16
512	32768	0.95	15
2	32768	244.14	15
512	16384	1.91	14
2	16384	488.28	14
512	8192	3.81	13
2	8192	976.56	13
512	4096	7.63	12
2	4096	1953.13	12
512	2048	15.26	11
2	2048	3906.25	11
512	512	31.04	9
2	512	15625.00	9
512	128	244.14	7
2	128	62500.00	7

NOTES:
1. This table is valid only if the DASM is connected to a free-running counter.

When using 16 bits of resolution on the comparator (MODE[2:0] = 000), the output can vary from a 0% duty cycle up to a duty cycle of 65535/65536. In this case it is not possible to have a 100% duty cycle. In cases where 16-bit resolution is not needed, it is possible to have a duty cycle ranging from 0% to 100%. Setting bit 15 of the value stored in register B to ‘1’ results in the output being ‘always set’. Clearing bit 15 (to ‘0’) allows normal comparisons to occur and the normal output waveform is obtained. Changes to and from the 100% duty cycle are done synchronously, as are all other width changes.

In the OPWM mode, the WOR bit selects whether the output is totem pole driven or open-drain.

9.5.3 DASM interrupts

When the FLAG bit is set, an interrupt request is generated on one of eight levels as defined by the interrupt level bits (IL[2:0]) in the DASMSIC register. If the interrupt level is set to zero, interrupts are disabled.

9.5.4 Freeze Action on the DASM

When the IMB FREEZE signal is recognized, the DASM capture and compare functions are halted. As soon as the FREEZE signal is negated, DASM actions resume as if nothing had happened. During freeze, the IN bit of the DASMSIC register is readable

and returns the level present at the input pin if an input mode is selected, or the output value if an output mode is in operation. When one of the output modes is in operation, the force output function remains available, allowing the software to output the desired level and simplifying debugging. All DASM registers are accessible during freeze.

9.5.5 DASM Registers

The DASM register map comprises four 16-bit register locations. As shown in **Table 9-10**, the register block contains three DASM registers and one reserved register. All unused bits and reserved address locations return zero when read by the software. Writing to unused bits and reserved address locations has no meaning or effect. All register addresses in this section are specified as offsets from the base address of the DASM. In CTM implementations featuring multiple DASMs, each DASM has its own set of registers.

Table 9-10 DASM Register Map

Address	15 8	7 0
0xYF F718	DASM3 status/interrupt/control register (DASM3SIC)	
0xYF F71A	DASM3 register A (DASM3A)	
0xYF F71C	DASM3 register B (DASM3B)	
0xYF F720	DASM4 status/interrupt/control register (DASM4SIC)	
0xYF F722	DASM4 register A (DASM4A)	
0xYF F724	DASM4 register B (DASM4B)	
0xYF F748	DASM9 status/interrupt/control register (DASM9SIC)	
0xYF F74A	DASM9 register A (DASM9A)	
0xYF F74C	DASM9 register B (DASM9)	
0xYF F750	DASM10 status/interrupt/control register (DASM10SIC)	
0xYF F752	DASM10 register A (DASM10A)	
0xYF F754	DASM10 register B (DASM10)	

9.5.5.1 DASMSIC — DASM Status/Interrupt Control Register

DASM3SIC — DASM Status/Interrupt Control Register **0xYF F718**
DASM4SIC **0xYF F720**
DASM9SIC **0xYF F748**
DASM10SIC **0xYF F750**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
FLAG	IL2	IL1	IL0	IARB3	0	WOR	BSL	IN	FORCA	FORCB	ED-POL	MODE 3	MODE 2	MODE 1	MODE 0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-11 DASMSIC Bit Settings

Bit(s)	Name	Description
15	FLAG	Flag status. This status bit indicates whether or not an input capture or output compare event has occurred. If the IL field is non-zero, an interrupt request is generated when the FLAG bit is set. The flag clearing mechanism will work only if no flag setting event occurs between the read and write operations; if a FLAG setting event occurs between the read and write operations, the FLAG bit will not be cleared. – In the DIS mode, the FLAG bit is cleared. – In the IPWM mode, the FLAG bit is set each time there is a capture on channel A. – In the IPM mode, the FLAG bit is set each time there is a capture on channel A, except for the first time. – In the IC mode, the FLAG bit is set each time there is a capture on channel A. – In the OCB mode (i.e. when MODE0 = 0), the FLAG bit is only set each time there is a successful comparison on channel B. In the OCAB mode (i.e. when MODE0 = 1), the FLAG bit is set each time there is a successful comparison on either channel A or B. – In the OPWM mode, the FLAG bit is set whenever there is a successful comparison on channel A. 0 = An input capture or output compare event has not occurred. 1 = An input capture or output compare event has occurred.
14:12	IL[2:0]	Interrupt level. The three interrupt level bits are read/write control bits that select the priority level of interrupt requests made by the DASM. These bits can be read or written at any time and are cleared by reset. 000 = Interrupt disabled 001 = Interrupt level 1 (lowest) 010 = Interrupt level 2 011 = Interrupt level 3 100 = Interrupt level 4 101 = Interrupt level 5 110 = Interrupt level 6 111 = Interrupt level 7 (highest)
11	IARB3	Interrupt arbitration bit 3. The read/write IARB3 bit works in conjunction with the IARB[2:0] field in the BIUSM module configuration register. Each module that generates interrupt requests on the IMB must have a unique value in the arbitration field (IARB). This interrupt arbitration identification number is used to arbitrate for the IMB when modules generate simultaneous interrupts of the same priority.
10	—	Reserved
9	WOR	Wired-OR. In the DIS, IPWM, IPM and IC modes, the WOR bit is not used; reading this bit returns the value that was previously written. In the OCB, OCAB and OPWM modes, the WOR bit selects whether the output buffer is configured for open-drain or totem pole operation. 0 = Output buffer is totem pole. 1 = Output buffer is open-drain.
8	BSL	Bus select. This control bit selects the time base bus to be connected to the DASM. 0 = The DASM is connected to time base bus A. 1 = The DASM is connected to time base bus B.
7	IN	Input pin status. In the DIS, IPWM, IPM and IC modes, this read-only status bit reflects the logic level on the input pin. In the OCB, OCAB and OPWM modes, reading this bit returns the value latched on the output flip-flop, after EDPOL polarity selection. Writing to this bit has no effect.
6:5	FORCA, FORCB	Force A, B. In the OCB, OCAB and OPWM modes, the FORCA, B bit allows the software to force the output flip-flop to behave as if a successful comparison had occurred on channel A, B (except that the FLAG bit is not set). Writing a one to FORCA, B sets the output flip-flop; writing a zero to it has no effect. In the DIS, IPWM, IPM and IC modes, FORCA and FORCB are not used and writing to them has no effect. Writing a one to both FORCA and FORCB simultaneously resets the output flip-flop

Table 9-11 DASMSIC Bit Settings (Continued)

Bit(s)	Name	Description
4	EDPOL	Edge polarity. In the DIS mode, this bit is not used; reading it returns the last value written. In the IPWM mode, this bit is used to select the capture edge sensitivity of channels A and B. 0 = Channel A captures on a rising edge. Channel B captures on a falling edge. 1 = Channel A captures on a falling edge. Channel B captures on a rising edge. In the IPM and IC modes, the EDPOL bit is used to select the input capture edge sensitivity of channel A. 0 = Channel A captures on a rising edge. 1 = Channel A captures on a rising edge. In the OCB, OCAB and OPWM modes, the EDPOL bit is used to select the voltage level on the output pin. 0 = The output flip-flop logic level appears on the output pin: a compare on channel A sets the output pin, a compare on channel B resets the output pin. 1 = The complement of the output flip-flop logic level appears on the output pin: a compare on channel A resets the output pin; a compare on channel B sets the output pin.
3:0	MODE[3:0]	Mode select. The four mode select bits select the mode of operation of the DASM. To avoid spurious interrupts, it is recommended that DASM interrupts are disabled before changing the operating mode.

9.5.5.2 DASMA — DASM Data Register A

DASM3A — DASM Data Register A **0xYF F71A**
DASM4A **0xYF F722**
DASM9A **0xYF F74A**
DASM10A **0xYF F752**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MSB								LSB							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DASMA is the data register associated with channel A; its use varies with the different modes of operation:

- In the DIS mode, DASMA can be accessed to prepare a value for a subsequent mode selection.
- In the IPWM mode, DASMA contains the captured value corresponding to the trailing edge of the measured pulse.
- In the IPM and IC modes, DASMA contains the captured value corresponding to the most recently detected dedicated edge (rising or falling edge).
- In the OCB and OCAB modes, DASMA is loaded with the value corresponding to the leading edge of the pulse to be generated. Writing to DASMA in the OCB and OCAB modes also enables the corresponding channel A comparator until the next successful comparison.
- In the OPWM mode, DASMA is loaded with the value corresponding to the leading edge of the PWM pulse to be generated.

9.5.5.3 DASMB — DASM Data Register B

DASM3B — DASM Data Register B **0xYF F71C**
DASM4B **0xYF F724**
DASM9B **0xYF F74C**
DASM10B **0xYF F754**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MSB								LSB							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DASMB is the data register associated with channel B; its use varies with the different modes of operation. Depending on the mode selected, software access is to register B1 or register B2.

In the DIS mode, DASMB can be accessed to prepare a value for a subsequent mode selection. In this mode, register B1 is accessed in order to prepare a value for the OPWM mode. Unused register B2 is hidden and cannot be read, but is written with the same value when register B1 is written.

In the IPWM mode, DASMB contains the captured value corresponding to the leading edge of the measured pulse. In this mode, register B2 is accessed; buffer register B1 is hidden and cannot be accessed.

In the IPM and IC modes, DASMB contains the captured value corresponding to the most recently detected period edge (rising or falling edge). In this mode, register B2 is accessed; buffer register B1 is hidden and cannot be accessed.

In the OCB and OCAB modes, DASMB is loaded with the value corresponding to the trailing edge of the pulse to be generated. Writing to DASMB in the OCB and OCAB modes also enables the corresponding channel B comparator until the next successful comparison. In this mode, register B2 is accessed; buffer register B1 is hidden and cannot be accessed.

In the OPWM mode, DASMB is loaded with the value corresponding to the trailing edge of the PWM pulse to be generated. In this mode, register B1 is accessed; buffer register B2 is hidden and cannot be accessed.

9.6 Pulse Width Modulation Submodule (PWMSM)

The purpose of the pulse width modulation submodule (PWMSM) is to create a variable pulse width output signal at a wide range of frequencies, independent of other CTM9 output signals. The PWMSM includes its own counter, and thus does not use the CTM9 time-base buses. The PWMSM pulse width can vary from 0.0 percent to 100.0 percent, with up to 16 bits of resolution. The finest output resolution is the MCU system clock time divided by two (for a system clock of 16.78 MHz, the finest output pulse width resolution is 119 nanoseconds). With the full 16 bits of resolution and the first stage prescaler divide-by-2 clock selection, the period of the PWM output can

range from 7.8 milliseconds to 2.0 seconds (assuming a 16.78 MHz MCU clock). By reducing the counting value, the output signal period can be reduced. The period can be as fast as 488 microseconds (2.048 KHz) with 12 bits of resolution, as fast as 30.5 microseconds (32.768 KHz) with 8 bits of resolution, and as fast as 7.6 microseconds (131.072 KHz) with 6 bits of resolution (still assuming a 16.78 MHz system clock and a first stage prescaler divide-by-2 clock selection). A block diagram of the PWMSM is shown in **Figure 9-13**.

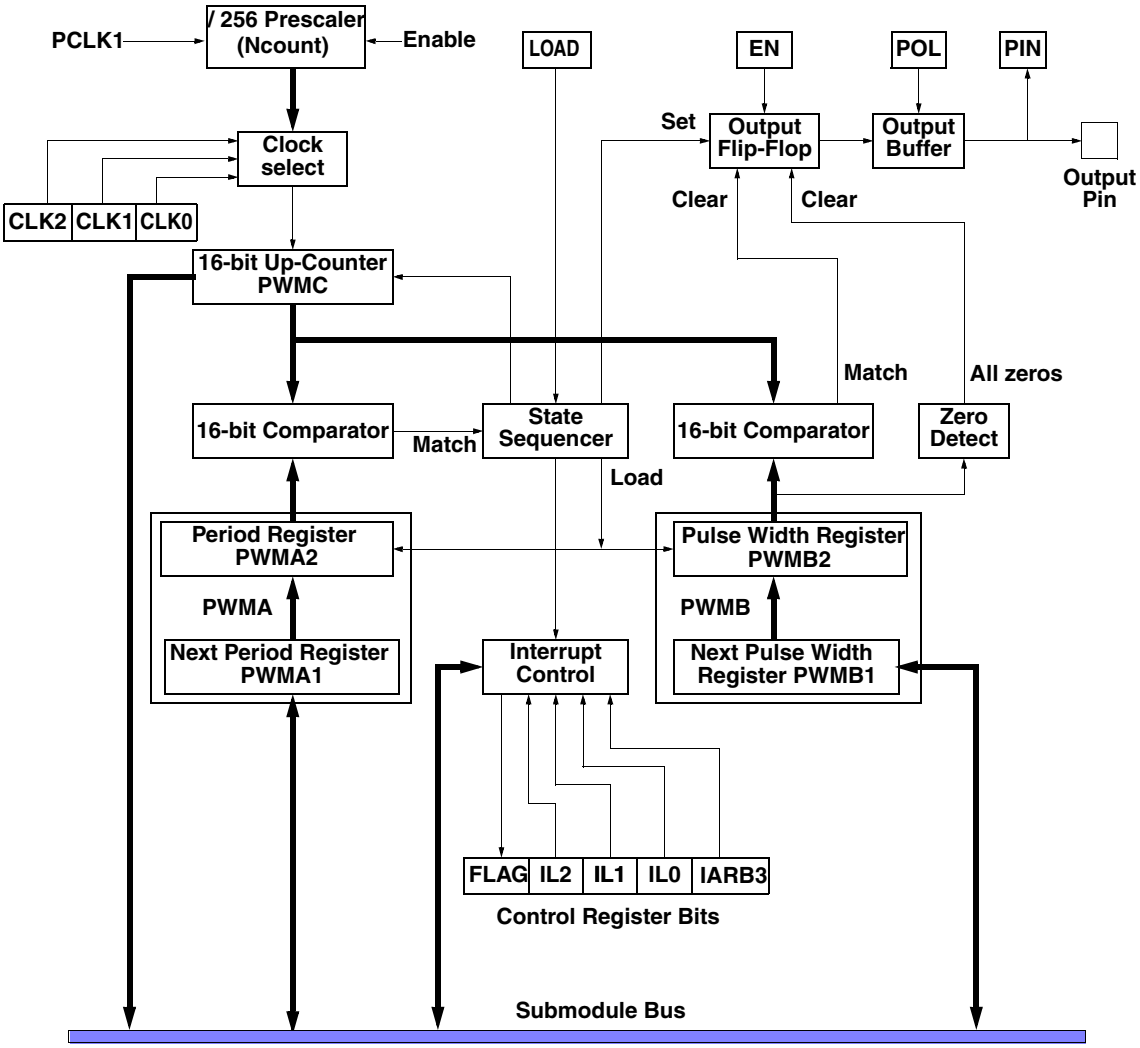


Figure 9-13 Pulse Width Modulation Submodule Block Diagram

9.6.1 Output Flip-Flop and Pin

The output flip-flop is the basic output mechanism of the PWMSM. Except when the required pulse width is 0% or 100%, the output flip-flop is set at the beginning of each period and is cleared at the end of the designated pulse width. The polarity of the output pulse can be selected in software. The output of the PWMSM is connected to an external, output-only pin. When the PWMSM is not required, and is disabled by clear-

ing the EN bit in the PWMSIC register, this pin serves as a digital output-only port pin. When the PWMSM is disabled, the POL bit in the SIC register serves as an output port bit.

9.7 Time Base Bus System

The time base bus (TBB) system makes it possible to freely configure connections between counter submodules and action submodules. However the PWMSM submodules are independent of the time base bus system. The CTM9 configuration is shown in **Figure 9-1**.

9.7.1 Clock Selection

The PWMSM contains an 8-bit prescaler that is clocked by the PCLK1 signal from the CPSM (i.e., the MCU system clock divided by two or by three). A 3-bit field (CLK[2:0]) in the PWMSM status, interrupt and control register (PWMSIC) allows the software to select which of the 8 prescaler outputs drives the PWMSM counter. The prescaler outputs are the main MCU clock divided by: 2, 4, 8, 16, 32, 64, 128 and 512 (or 3, 6, 12, 24, 48, 96, 192 and 768, if the divide-by-3 option is used in the CPSM to generate PCLK1).

9.7.2 The PWMSM Counter (PWMC)

The 16-bit up-counter in the PWMSM provides the time base for the PWM output signal. The counter is held in the 0x0001 state on reset or when the PWMSM is disabled. When the PWMSM is enabled, the counter begins counting at the rate defined by the clock selection. Each time the counter matches the contents of the period register, the counter is preset to 0x0001 and starts to count from that value. The counter can be read at any time without affecting its value. Writing to the counter has no effect.

9.7.3 PWMSM Period Registers and Comparator

The period section of the PWMSM consists of two 16-bit period registers (PWMA1 and PWMA2) and one 16-bit comparator. PWMA2 holds the current PWM period value and PWMA1 holds the next PWM period value. The software establishes the next period of the output PWM signal by writing a value into PWMA1. PWMA2 acts as a double buffer of PWMA1, allowing the contents of PWMA1 to be changed at any time without affecting the current period of the output signal; it cannot be accessed directly by the software. PWMA1 can be read or written at any time. The new value in the PWMA1 register is transferred to PWMA2 on the next full cycle of the output or when a ‘1’ is written to the LOAD bit in the PWMSIC register.

The comparator continuously compares the contents of the PWMA2 register with the value in the PWMSM counter. When a match occurs, the state sequencer sets the output flip-flop and resets the counter to 0x0001.

Period values 0x0000 and 0x0001 are special cases. When PWMA2 contains 0x0000, an output period of 65536 PWM clock periods is generated.

When PWMA2 contains 0x0001, a period match occurs on every PWM clock period: the counter never increments beyond 0x0001 and the output level never changes.

NOTE

A value of 0x0002 in the period register and a value of 0x0001 in the pulse register are the conditions necessary to obtain the maximum possible output frequency for a given PWM clock period.

9.7.4 PWMSM Pulse Width Registers and Comparator

The pulse width section of the PWMSM consists of two 16-bit pulse width registers (PWMB1 and PWMB2) and one 16-bit comparator. PWMB2 holds the current PWM pulse width value and PWMB1 holds the next PWM pulse width value. The software establishes the next pulse width of the output PWM signal by writing a value into PWMB1. Software may write a new pulse width value into PWMB1 at any time and this new value will take effect at the start of the next PWM period (or when the LOAD bit in the PWMSIC register is written to a '1'). The PWMSM hardware does not modify the contents of PWMB1 at any time.

PWMB2 acts as a double buffer of PWMB1, allowing the contents of PWMB1 to be changed at any time without affecting the current pulse width of the output signal; it cannot be accessed directly by the software. PWMB1 can be read or written at any time. The new value in the PWMB1 register is transferred to PWMB2 on the next full cycle of the output or when a '1' is written to the LOAD bit in the PWM SIC register.

The pulse width comparator is a 16-bit 'ones-equality' comparator that compares the contents of the PWMB2 register with the 16-bit PWM counter. When the counter reaches the value in PWMB2, a match occurs and the output flip-flop is cleared. This pulse width match completes the pulse width; it does not affect the counter. Since a 'ones-equality' comparator is used, subsequent comparisons can occur, but will have no effect on the output signal as the output flip-flop has already been cleared.

The PWM output pulse may be as short as one PWM clock period (PWMB2 = 0x0001). It may be as long as one PWM clock period less than the PWM period; for example, the pulse width equal to 65535 PWM clock periods can be obtained by setting PWMB2 = 0xFFFF and PWMA2 = 0x0000.

9.7.5 0% and 100% 'Pulses'

The 0% and 100% 'pulses' are special limiting cases (zero width and infinite width) that are defined by the 'always clear' and 'always set' states of the output flip-flop.

The 0% pulse is generated by making the pulse width value in PWMB2 equal to 0x0000. The output is a true steady state signal with no glitches.

The 100% pulse is created by making the pulse width value in PWMB2 equal to or greater than the period value in PWMA2. The output is a true steady state signal with no glitches.

It is not possible to have a 100% duty cycle when the output period is selected to be 65536 PWM clock periods (by setting PWMB2 = 0x0000); in this case the maximum duty cycle is 99.998% (100 x 65535/65536).

When using the PWM output signal to generate analog levels, the 0% and 100% pulses provide the full scale values.

Even when 0% or 100% pulses are being generated, the 16-bit PWM counter continues to count and output changes to or from these limit values are done synchronously with the selected period.

9.7.6 PWMSM Coherency

Byte access of registers is discussed in **9.5.1 32-Bit Coherent Access**, however, it should be noted that byte writes to the double buffered registers PWMA1 and PWMB1 are not recommended as the transfer from the primary registers to the secondary registers is done on a word basis.

For most PWMSM operations, 16-bit accesses are sufficient and long word accesses are treated as two word accesses, with one exception — a long word write to the period/pulse width registers. In this case, if the long word write is done within the PWM period, there is no visible effect on the output signal and the new values are stored in PWMA1 and PWMB1 ready to be loaded into the buffer registers at the start of the next period. If, however, the long word write coincides with the end of the period, then the transfer of values from the primary registers to the secondary registers is suppressed until the end of the next PWM period; during this period, the current values in the secondary registers are used for the period and the pulse width.

9.7.7 PWMSM Interrupts

The FLAG bit in the PWMSIC register is set when a new period begins and indicates that the period and pulse width registers (PWMA1 and PWMB1) may be updated with values for the next output period and pulse width. When the FLAG bit is set, an interrupt request is generated on one of eight levels as defined by the interrupt level bits (IL[2:0]) in the PWMSIC register. If the interrupt level is set to zero, interrupts are disabled.

9.7.8 Freeze Action on the PWMSM

When the IMB FREEZE signal is recognized, the PWMSM counter stops incrementing and remains set at its last value. When the FREEZE signal is negated, the counter starts incrementing from its last value, as if nothing had happened.

9.7.9 PWM frequency, Pulse Width and Resolution

Table 9-12 and **Table 9-13** shows the pulse widths and frequencies that can be achieved using the /2 and /3 options and a clock frequency of 16.78 MHz.

Table 9-12 PWM Pulse and Frequency Ranges (in Hz) Using /2 Option (16.78 MHz)

Minimum Pulse Width	Bits of Resolution															
	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0.119µs/2	128	256	512	1024	2048	4096	8192	16384	32768	65.5K	131K	262K	524K	1049K	2097K	4195K
0.238µs/4	64	128	256	512	1024	2048	4096	8192	16384	32768	65.5K	131K	262K	524K	1049K	2097K
0.477µs/8	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65.5K	131K	262K	524K	1049K
0.954µs/16	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65.5K	131K	262K	524K
1.91µs/32	8.0	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65.5K	131K	262K
3.81µs/64	4.0	8.0	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65.5K	131K
7.63µs/128	2.0	4.0	8.0	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65.5K
30.5µs/512	0.5	1.0	2.0	4.0	8.0	16	32	64	128	256	512	1024	2048	4096	8192	16384

Table 9-13 PWM Pulse and Frequency Ranges (in Hz) Using /3 Option (16.78 MHz)

Minimum Pulse Width	Bits of Resolution															
	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0.179µs/3	85.33	170.7	341.3	682.7	1365	2731	5461	10923	21845	43.69K	87.38K	174.8K	349.5K	699.1K	1398K	2796K
0.358µs/6	42.67	85.33	170.7	341.3	682.7	1365	2731	5461	10923	21845	43.69K	87.38K	174.8K	349.5K	699.1K	1398K
0.715µs/12	21.33	42.67	85.33	170.7	341.3	682.7	1365	2731	5461	10923	21845	43.69K	87.38K	174.8K	349.5K	699.1K
1.431µs/24	10.67	21.33	42.67	85.33	170.7	341.3	682.7	1365	2731	5461	10923	21845	43.69K	87.38K	174.8K	349.5K
2.861µs/48	5.333	10.67	21.33	42.67	85.33	170.7	341.3	682.7	1365	2731	5461	10923	21845	43.69K	87.38K	174.8K
5.722µs/96	2.667	5.333	10.67	21.33	42.67	85.33	170.7	341.3	682.7	1365	2731	5461	10923	21845	43.69K	87.38K
11.44µs/192	1.333	2.667	5.333	10.67	21.33	42.67	85.33	170.7	341.3	682.7	1365	2731	5461	10923	21845	43.69K
45.78µs/768	0.333	0.667	1.333	2.667	5.333	10.67	21.33	42.67	85.33	170.7	341.3	682.7	1365	2731	5461	10923

9.7.10 PWM Frequency

The relationship between the PWM output frequency (f_{PWMO}) and the MCU system clock frequency (f_{SYS}) is given by Equation .

$$f_{PWMO} = \frac{f_{SYS}}{N_{CLOCK} \bullet N_{COUNTER}}$$

where N_{CLOCK} is the CPSM clock divide ratio (2 or 3) and $N_{COUNTER}$ is the PWMSM counter divide ratio.

9.7.11 PWM Pulse Width

The minimum output pulse width (t_{PWMIN}) and the MCU system clock frequency (f_{SYS}) is given by Equation .

$$t_{PWMIN} = \frac{N_{CLOCK}}{f_{SYS}}$$

9.7.12 PWM Period and Pulse Width Register Values

The value to be loaded into the PWM period register (PWMA1) to obtain a given period is given by Equation .

$$PWMA1 = \frac{f_{SYS}}{N_{CLOCK} \bullet f_{PWMO}}$$

The value to be loaded into the PWM pulse width register (PWMB1) to obtain a given period is given by Equation .

$$PWMB1 = \frac{t_{PWMO}}{t_{PWMIN}} = \frac{\text{Duty cycle \%}}{100} \cdot PWMA1$$

where $t_{(PWMO)}$ is the actual output pulse width.

9.7.13 PWMSM Register Map and Registers

The PWMSM register map comprises four 16-bit registers as shown in **Table 9-14**. All unused bits and reserved address locations return zero when read by the software. Writing to unused bits and reserved address locations has no meaning nor effect. All register addresses in this section are specified as offsets from the base address of the PWMSM.

Table 9-14 PWMSM Register Map

Address	15 8	7 0
0xYF F728	PWM5 Status, interrupt and control register (PWM5SIC)	
0xYF F72A	PWM5 period register(PWM5A)	
0xYF F72C	PWM5 pulse width register (PWM5B)	
0xYF F72E,	PWM5 counter register (PWM5C)	
0xYF F730	PWM6 Status, interrupt and control register (PWM6SIC)	
0xYF F732	PWM6 period register(PWM6A)	
0xYF F734	PWM6 pulse width register (PWM6B)	
0xYF F736	PWM6 counter register (PWM6C)	
0xYF F738	PWM7 Status, interrupt and control register (PWM7SIC)	
0xYF F73A	PWM7 period register(PWM7A)	
0xYF F73C	PWM7 pulse width register (PWM7B)	
0xYF F73E	PWM7 counter register (PWM7C)	
0xYF F740	PWM8 Status, interrupt and control register (PWM8SIC)	
0xYF F742	PWM8 period register(PWM8A)	
0xYF F744	PWM8 pulse width register (PWM8B)	
0xYF F746	PWM8 counter register (PWM8C)	

9.7.13.1 PWMSIC — Status, Interrupt and Control Register

The PWMSIC register contains status, interrupt enable and control bits for the PWMSM. It also contains interrupt level and arbitration bits.

PWM5SIC — PWM Status/Interrupt Control Register **0xYF F728**
PWM6SIC **0xYF F730**
PWM7SIC **0xYF F738**
PWM8SIC **0xYF F740**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
FLAG	IL2	IL1	IL0	IARB3	0	0	0	PIN	0	LOAD	POL	EN	CLK2	CLK1	CLK0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-15 PWMSIC Bit Settings

Bit(s)	Name	Description
15	FLAG	Period completion status. The FLAG bit is a status bit that indicates when the PWM output period has been completed. The FLAG bit is set by the hardware each time a PWM period is completed. Whenever the PWM is enabled, the FLAG bit is set immediately to indicate that the contents of the buffer registers PWMA2 and PWMB2 have been updated, and that the period using these new values has started. It also indicates that the user accessible period and pulse width registers PWMA1 and PWMB1 can be loaded with values for the next PWM period. Once set, the FLAG bit will remain set and will not be affected by any subsequent period completions, until it is cleared by the software. The FLAG bit can only be cleared by software. To clear the flag, the software must first read the bit (as 'one') then write a 'zero' to the bit. Writing a one to the FLAG bit has no effect. When the PWM is disabled the FLAG bit remains in the cleared state. The flag clearing mechanism will work only if no flag setting event occurs between the read and write operations; if a FLAG setting event occurs between the read and write operations, the FLAG bit will not be cleared. When the interrupt level set by the interrupt level bit IL[2:0] is not equal to zero, an interrupt request is generated when the FLAG bit is set. Before returning from the interrupt service routine, the FLAG bit should be cleared by software to prevent the PWMSM from immediately generating another interrupt request on the IMB. 0 = PWM period not completed. 1 = PWM period completed.
14:12	IL[2:0]	Interrupt level. The three interrupt level bits select the interrupt level of requests made by the PWMSMt. 000 = Interrupt disabled 001 = Interrupt level 1 (lowest) 010 = Interrupt level 2 011 = Interrupt level 3 100 = Interrupt level 4 101 = Interrupt level 5 110 = Interrupt level 6 111 = Interrupt level 7 (highest)
11	IARB3	Interrupt arbitration. The read/write IARB3 bit works in conjunction with the IARB[2:0] field in the BIUSM module configuration register. Each module that generates interrupt requests on the IMB must have a unique value in the arbitration field (IARB). This interrupt arbitration identification number is used to arbitrate for the IMB when modules generate simultaneous interrupts of the same priority. The IARB3 bit is cleared by reset.
10:8	—	Reserved
7	PIN	Output pin status. The PIN bit is a status bit that indicates the logic state present on the output pin. The software can thus monitor the waveform being created on the output pin. PIN is a read-only bit; writing to it has no effect. 0 = Logic zero state present on the output pin. 1 = Logic one state present on the output pin.
6	—	Reserved

Table 9-15 PWMSIC Bit Settings (Continued)

Bit(s)	Name	Description
5	LOAD	Load control. The LOAD bit is a control bit that allows the software to reinitialize the PWMSM and start a new PWM period without causing a glitch on the PWM output signal. 0 = No action. 1 = Load period and pulse width registers. This bit is always read as a zero. Writing a one to this bit results in the following immediate actions: – The contents of PWMA1 (period value) are transferred to PWMA2, – The contents of PWMB1 (pulse width value) are transferred to PWMB2, – The counter register (PWMC) is initialized to 0x0001, – The control logic and state sequencer are reset, – The FLAG bit is set, and – The output flip-flop is set if the new value in PWMB2 is different from 0x0000.
4	POL	Output pin polarity control. The POL bit is a control bit that allows the software to set the polarity of the PWM output signal. It works in conjunction with the EN bit and controls whether the PWMSM drives the output pin with the true or inverted value of the output flip-flop, see **Table 9-16**.
3	EN	Enable control. The EN bit is a control bit that allows the software to enable and disable the PWMSM as required. 0 = Disable the PWMSM and stop generation of PWM output pulses. 1 = Enable the PWMSM and start generation of PWM output pulses. While the PWMSM is disabled (EN = 0): – The output flip-flop is held reset and the level on the output pin is set to one or zero according to the state of the POL bit, – The PWMSM's divide-by-256 prescaler is held in reset, – The counter stops incrementing and is held equal to 0x0001, – The comparators are disabled, – And the PWMA1 and PWMB1 registers permanently transfer their contents to the buffer registers (PWMA2 and PWMB2, respectively). When the EN bit is changed from zero to one: – The output flip-flop is set to start the first pulse, – The PWMSM's divide-by-256 prescaler is released, – The counter is released and starts to increment from 0x0001, – And the FLAG bit is set (to indicate that PWMA1 and PWMB1 can be updated with new values of period and pulse width. While EN is set, the PWMSM generates continuously a pulse width modulated output signal based on the data in PWMA2 and PWMB2 (which are updated via PWMA1 and PWMB2 each time a period is completed). To prevent unwanted glitches on the output waveform when disabling the PWMSM, the EN bit should not be cleared by the software until one period has been output as a 0% pulse (PWMB2 = 0x0000)
2:0	CLK[2:0]	Clock rate selection. The CLK bits are control bits that allow the software to select one of the eight counter clock sources coming from the PWMSM prescaler. These bits can be changed by the software at any time. **Table 9-17** shows the counter clock sources and rates in detail.

Table 9-16 PWMSM Output Pin Polarity Selection

Control Bits		Output Pin State	Periodic Edge	Variable Edge	Optional Interrupt On
POL	EN				
0	0	Always low	—	—	—
1	0	Always high	—	—	—
0	1	High pulse	Rising edge	Falling edge	Rising edge
1	1	Low pulse	Falling edge	Rising edge	Falling edge

Table 9-17 PWMSM Clock Rate Selection

PWMSM CLK Bits			CPSM Bit DIV23	PWMSM Clock	Clock Source
CLK2	CLK1	CLK0			
0	0	0	0	f_{SYS} / 2	PCLK1
0	0	1	0	f_{SYS} / 4	Prescaler (/2)
0	1	0	0	f_{SYS} / 8	Prescaler (/4)
0	1	1	0	f_{SYS} / 16	Prescaler (/8)
1	0	0	0	f_{SYS} / 32	Prescaler (/16)
1	0	1	0	f_{SYS} / 64	Prescaler (/32)
1	1	0	0	f_{SYS} / 128	Prescaler (/64)
1	1	1	0	f_{SYS} / 512	Prescaler (/256)
0	0	0	1	f_{SYS} / 3	PCLK1
0	0	1	1	f_{SYS} / 6	Prescaler (/2)
0	1	0	1	f_{SYS} / 12	Prescaler (/4)
0	1	1	1	f_{SYS} / 24	Prescaler (/8)
1	0	0	1	f_{SYS} / 48	Prescaler (/16)
1	0	1	1	f_{SYS} / 96	Prescaler (/32)
1	1	0	1	f_{SYS} / 192	Prescaler (/64)
1	1	1	1	f_{SYS} / 768	Prescaler (/256)

9.7.13.2 PWMA — PWM Period Register

The PWMA register contains the period value for the next cycle of the PWM output waveform. In normal usage, with the PWMSM enabled, the software writes a period value into PWMA1 and this value is then loaded into the PWMA2 register at the end of the current period. If the PWMSM is disabled, a period value written to PWMA1 is loaded into PWMA2 on the next tic (of the MCU system clock). PWMA2 is a temporary register that is used for smoothly updating the PWM period value; it cannot be read or written directly by software.

Software may write a new period value into PWMA1 at any time and this new value will take effect at the start of the next PWM period (or when the LOAD bit in the PWMSIC register is written to a ‘1’). The PWMSM hardware does not modify the contents of PWMA1 at any time.

PWM5A — PWM Period Register **0xYF F72A**
PWM6A **0xYF F732**
PWM7A **0xYF F73A**
PWM8A **0xYF F742**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MSB								LSB							
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

9.7.13.3 PWMB — PWM Pulse Width Register

The PWMB register contains the pulse width value for the next cycle of the PWM output waveform. In normal usage, with the PWMSM enabled, the software writes a pulse width value into PWMB1 and this value is then loaded into the PWMB2 register at the end of the current period. If the PWMSM is disabled, a pulse width value written to PWMB1 is loaded into PWMB2 on the next tic (of the MCU system clock). PWMB2 is a temporary register that is used for smoothly updating the PWM pulse width value; it cannot be read or written directly by software.

Software may write a new pulse width value into PWMB at any time and this new value will take effect at the start of the next PWM period (or when the LOAD bit in the PWMSIC register is written to a ‘1’). The PWMSM hardware does not modify the contents of PWMB1 at any time.

PWM5B — PWM Pulse Width Register **0xYF F72C**
PWM6B **0xYF F734**
PWM7B **0xYF F73C**
PWM8B **0xYF F744**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MSB								LSB							
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

9.7.13.4 PWMC — PWM Counter Register

The counter (register PWMC) is read-only: software may read the counter register at any time; writing to it has no effect. PWMC is loaded with the value 0x0001 on reset and is set to that value and held whenever the PWMSM is disabled (EN = 0).

PWM5C — PWM Counter Register **0xYF F72E**
PWM6C **0xYF F736**
PWM7C **0xYF F73E**
PWM8C **0xYF F746**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MSB								LSB							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

9.8 Bus Interface Unit Submodule (BIUSM)

The bus interface unit submodule (BIUSM) allows all the CTM9 submodules to communicate to the IMB3 via the SMB (sub module bus).

9.8.1 Freeze Action on the BIUSM

When the IMB freeze condition is detected, the FRZ bit in the BIUSM module configuration register determines whether or not the freeze condition is passed on to the other CTM submodules. If FRZ = 0, the freeze condition is ignored; if FRZ = 1, the BIUSM passes the FREEZE signal from the IMB through to the CTM submodules. Each CTM submodule then reacts to the FREEZE signal as defined by its own internal circuitry and control bits.

9.8.2 LPSTOP Action on the BIUSM

When the CPU is stopped by an LPSTOP instruction (from CPU32 or CPU16), the system clock (f_{SYS}) is stopped, thereby shutting down all dependent modules, including the CTM, until the low-power STOP mode is exited.

9.8.3 STOP and WAIT Action on the BIUSM

When the STOP instruction on CPU32 or the WAIT instruction on CPU16 is executed, only the CPU is stopped; the CTM continues to operate as normal. (To stop the CTM operation selectively, refer to the description of the STOP bit in **9.8.4.1 BIUMCR — BIUSM Module Configuration Register**).

9.8.4 BIUSM Registers

The BIUSM register map comprises four 16-bit register locations. As shown in **Table 9-18**, the register block contains the three BIUSM registers and one reserved register. The BIUSM register block always occupies the first four register locations in the CTM register space and cannot be relocated within the CTM structure. All unused bits and reserved address locations return zero when read by the software. Writing to unused bits and reserved address locations has no effect.

Table 9-18 BIUSM Register Map

Address	15 8	7 0
0xYF F700	BIUSM module configuration register (BIUMCR)	
0xYF F702	BIUSM test register (BIUTST)	
0xYF F704	BIUSM time base register (BIUTBR)	

9.8.4.1 BIUMCR — BIUSM Module Configuration Register

The BIUMCR register contains nine defined bits that allow the software to control five functions of the CTM: enabling/disabling of the module, response to FREEZE, vector base address, interrupt arbitration number and access to the time base buses (via the time base register).

BIUMCR — BIUSM Module Configuration Register **0xYF F700**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
STOP	FRZ	0	VECT7	VECT6	IARB2	IARB1	IARB0	0	0	TBRS1	0	0	0	0	TBRS0
RESET:															
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0

Table 9-19 BIUMCR Bit Settings

Bit(s)	Name	Description
15	STOP	Stop enable. The STOP bit, while asserted, activates the FREEZE signal on the SMB regardless of the state of the FREEZE signal on the IMB. This completely stops the operation of the CTM. Note that some submodules may validate this signal with internal enable bits. The BIUSM continues to operate to allow the CPU access to the submodule's registers. The SMB FREEZE signal remains active until reset or until the STOP bit is negated by the CPU (via the IMB). 0 = Allows operation of the CTM. 1 = Stops operation of the CTM.
14	FRZ	Freeze enable. The FRZ bit, while asserted, activates the FREEZE signal on the SMB when the FREEZE signal on the IMB is active. This completely stops the operation of the CTM. Note that some submodules may validate this signal with internal enable bits. The BIUSM continues to operate to allow the CPU access to the submodule's registers. The SMB FREEZE signal remains active until the FRZ bit is cleared or the IMB FREEZE signal is negated. 0 = Ignores the FREEZE signal on the IMB. 1 = Halts the CTM sub module when the FREEZE signal appears on the IMB.
13	—	Reserved
12:11	VECT[7:8]	Interrupt vector base number. The interrupt vector base number bits select the interrupt vector base number for the CTM. Of the 8 bits necessary for vector number definition, the six least significant bits are programmed by hardware on a submodule basis, while the two remaining bits are provided by VECT7 and VECT6. 00 = Vector base number 0x00. 01 = Vector base number 0x40. 10 = Vector base number 0x80. 11 = Vector base number 0xC0.

Table 9-19 BIUMCR Bit Settings (Continued)

Bit(s)	Name	Description
10:8	IARG[2:0]	Interrupt arbitration identification. The interrupt arbitration bit field (IARB), composed of IARB[2:0] in the BIUMCR and the IARB3 bit within each submodule, provides fifteen different arbitration identification numbers that can be used to arbitrate between interrupt requests occurring on the IMB with the same interrupt priority level. The IARB field defaults to zero on reset, thus preventing the module from arbitrating during an interrupt arbitration acknowledge cycle (IACK). If no IMB arbitration takes place during the IACK cycle the spurious interrupt vector is generated by the SIM (system integration module). This tells the system that the interrupt arbitration number has not been initialized. The seven levels of interrupt are the primary means by which interrupt priority is established. The 4-bit interrupt arbitration number is the secondary priority, allowing up to 15 requests at each primary level. During the IACK cycle the request with the highest arbitration number gets serviced (binary 1111 is the highest priority and binary 0001 is the lowest). Many IMB modules have one software assignable arbitration number for the whole module. The CTM allows two different arbitration numbers to be used by providing each submodule with its own IARB3 bit (which can be set or cleared in software). Once IARB[2:0] are assigned in the BIUSM, they apply to all CTM interrupt requests. Therefore, CTM submodule interrupts can be interleaved in priority with requests from other modules at the same interrupt level.
7:6	—	Reserved
5,0	TBRS1, TBRS0	Time base register bus select. These bits specify which time base bus is accessed when the time base register (BIUTBR) is read. 00 = Time base bus TBB1 01 = Time base bus TBB2 10 = Time base bus TBB3 11 = Time base bus TBB4
4:1	—	Reserved

9.8.4.2 BIUTBR — BIUSM Time Base Register

In normal operation, the BIUTBR is a read-only register used to read the value present on one of the time base buses. The time base bus being accessed is determined by TBRS1 and TBRS0 in the BIUMCR. Writing to the BIUTBR has no effect, except in certain test modes.

BIUTBR — BIUSM Time Base Register **0xYF F704**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MSB								LSB							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

9.9 Counter Prescaler Submodule (CPSM)

The counter prescaler submodule (CPSM) generates six different clock frequencies which can be used by any counter submodule. Five of these frequencies are derived from a fixed divider. The divide ratio of the last clock frequency is software selectable from a choice of four divide ratios. Note that this submodule is contained within the BIUSM. A block diagram of the CPSM is given in **Figure 9-14**. The clock division ratios available on PCLKx are also shown in the table in **9.9.2.1 CPCR — CPSM Control Register**. These clock signals are provided on the SMB and may be used by any or all CTM submodules.

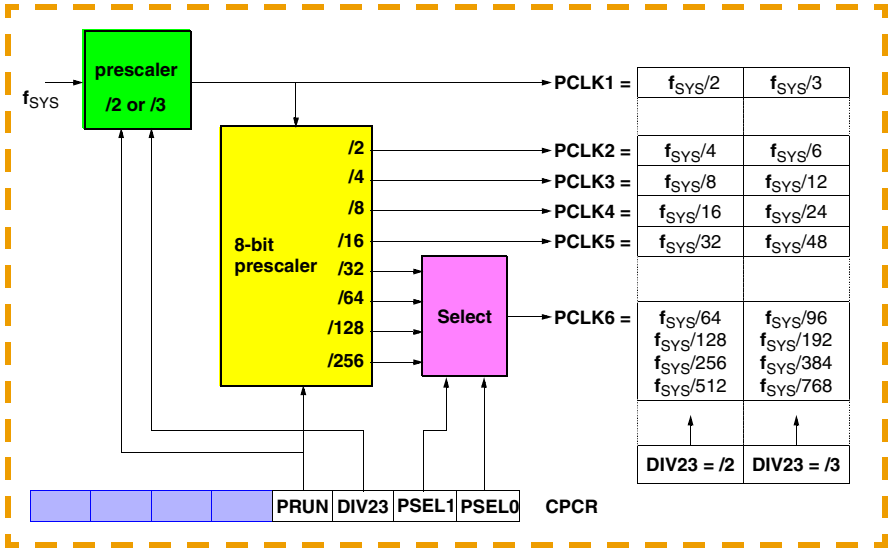


Figure 9-14 CPSM Block Diagram

9.9.1 Freeze Action on the CPSM

When the IMB FREEZE signal is recognized, the CPSM counters stop counting and remain set at their current values. When the FREEZE signal is negated, the counters start incrementing from their current values, as if nothing had happened. All registers are accessible during freeze.

9.9.2 CPSM Registers

The CPSM register map comprises four 16-bit register locations. As shown in **Table 9-20**, the register block contains two CPSM registers and two reserved registers. The CPSM register block always immediately follows the BIUSM register block in the CPSM register map. All unused bits and reserved address locations return zero when read by the software. Writing to unused bits and reserved address locations has no effect.

Table 9-20 CPSM Register Map

Address	15 8	7 0
0xYF F708	CPSM control register (CPCR)	
0xYF F70A	CPSM test register (CPTR)	

9.9.2.1 CPCR — CPSM Control Register

CPCR — CPSM Control Register **0xYF F708**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
0	0	0	0	0	0	0	0	0	0	0	0	PRUN	DIV23	PSEL 1	PSEL 0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-21 CPCR Bit Settings

Bit(s)	Name	Description
15:4	—	Reserved
3	PRUN	Prescaler running. The PRUN bit is a read/write control bit that allows the software to switch the prescaler counter on and off. This bit allows the counters in various CTM submodules to be synchronized. 0 = Prescaler divider is held in reset and is not running. 1 = Prescaler is running.
2	DIV23	Divide by 2 or divide by 3 . The DIV23 bit is a read/write control bit that selects the division ratio of the first prescaler counter. It may be changed by the software at any time and is cleared on reset. 0 = First prescaler stage divides by 2. 1 = First prescaler stage divides by 3.
1:0	PSEL[1:0]	Prescaler division ratio select. These control bits select the division ratio of the programmable prescaler output signal, PCLK6, See Table 9-22.

Table 9-22 Prescaler Division Ratio Select

Prescaler Control Register Bits				Prescaler Division Ratio					
PRUN	DIV23	PSEL1	PSEL0	PCLK1	PCLK2	PCLK3	PCLK4	PCLK5	PCLK6
0	X	X	X	0	0	0	0	0	0
1	0	0	0	2	4	8	16	32	64
1	0	0	1	2	4	8	16	32	128
1	0	1	0	2	4	8	16	32	256
1	0	1	1	2	4	8	16	32	512
1	1	0	0	3	6	12	24	48	96
1	1	0	1	3	6	12	24	48	192
1	1	1	0	3	6	12	24	48	384
1	1	1	1	3	6	12	24	48	768

9.9.3 Clock Sources for the Counter Submodules

The software chooses one of seven clock sources for each counter. Six of them are prescaler taps derived from the on-chip oscillator. The highest frequency available to the counter is the MCU system clock divided by two. Four of the other five taps are binary divisible from the system clock cycle — divide by 4, 8, 16, and 32. Another input clock to the counter is a software defined divide by 64, 128, 256, or 512 from the MCU clock. There is an alternate prescaler option where the MCU clock is divided by 3, 6, 12, 24, 48, 96, 192, 384, and 768. The seventh selectable clock source is an external

pin, which may trigger on the rising or falling edge of the input signal. The external input allows the counter to use a frequency not based on the microcontroller oscillator. An alternate use for the external clock source is for event or pulse counting.

9.10 CTM9 Interrupts

The CTM9 is able to generate a diverse set of interrupts on the IMB3. Each interrupting submodule is capable of requesting an interrupt on any of seven levels. A 3-bit level number and a 1-bit arbitration number included in each submodule are initialized by the software. The 3-bit level number selects which of the seven interrupt signals on the IMB are driven by that submodule to create an interrupt request. Of the four priority bits provided on the IMB3 during arbitration among the modules, one of them comes from the interrupting submodule and the CTM9 BIUSM provides the other three. Thus, the CTM9 may respond to two of the possible fifteen arbitration numbers.

During the IMB3 arbitration process, the CTM9 BIUSM manages the separate arbitration among the CTM9 submodules to determine which submodule will respond. Of the submodules which have an interrupt request pending at the level being arbitrated on the IMB, the submodule which has the lowest address is given the highest priority to respond.

Following the interrupt arbitration process, the CTM9 provides an 8-bit vector number. Six of the eight bits are provided by the interrupting submodule. Of the submodules produced to date, a submodule can identify up to two separate interrupt causes, each with unique interrupt vectors. The high-order two bits of the 8-bit vector are provided by the CTM9 BIUSM. The low order six vector bits identify the highest priority interrupt request pending in the CTM9 at the beginning of the arbitration cycle.

9.11 CTM9 Function Examples

The versatility of the CTM9 timer architecture is based on multiple counters and capture/compare channel units interconnected on time-base buses. Rather than present block diagrams of each submodule, this section includes some typical application examples — to show how the submodules can be interconnected to form timing functions. The diagrams used to illustrate these examples show only the blocks utilized for that function.

To illustrate the timing range of the CTM9 in different applications, many of the following paragraphs include time intervals quoted in microseconds and seconds. The assumptions used are that the microcontroller system clock is at 16.78 MHz with minimum prescaling (0.119 microsecond cycle) and with the maximum prescaling (48.0 microsecond cycle). For other system clock cycle rates and prescaler choices, the times mentioned in these paragraphs scale appropriately.

9.11.1 CTM9 Single Input Capture

The CTM9 single-action submodule (SASM) has an input capture register to latch the current state of a time-base bus when an external input edge is detected. The SASM is software programmable to latch on the rising or falling edge of the input signal. The software also selects one of two time-base buses, each originating at a counter sub-

module. The software can also enable an interrupt to occur when the input edge is detected to notify the software that new edge capture information is available.

Figure 9-15 shows an example of an MCSM a counter submodule for an SASM configured for input capturing. To measure the period of an incoming signal, the software reads and saves the latched value in register A for one edge, then when the next edge arrives, the software subtracts the new captured value in register A from the previously saved value to obtain the period interval. The maximum period that can be measured is the worst case software response time to a newly captured value.

The software measures the width of a pulse in a very similar way, the only difference is that after each edge, the edge detector is reprogrammed to trigger on the next opposite edge.

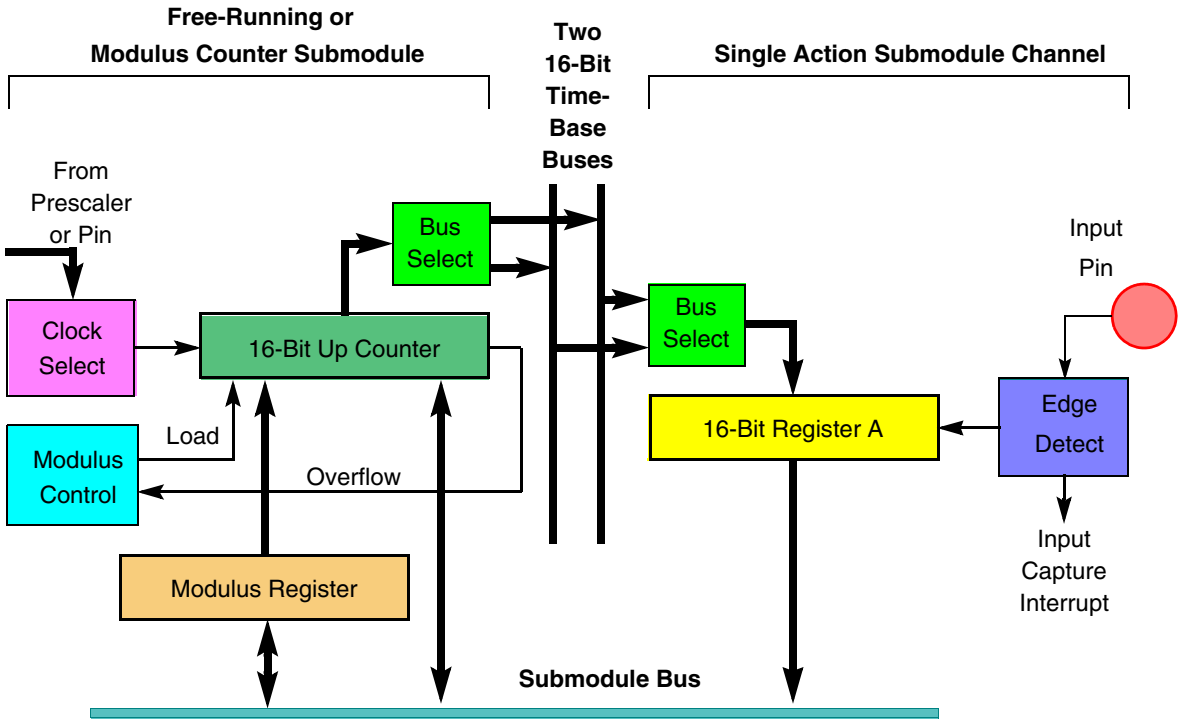


Figure 9-15 CTM9 Example — Single Edge Input Capture

9.11.2 CTM9 Input Double Edge Pulse Width Measurement

To measure the width of an input pulse, the CTM9 double-action submodule (DASM) has two capture registers so that only one interrupt is needed after the second edge. The software can read both edge samples and subtract them to get the pulse width. The leading edge sample is double latched so that the software has the time of one full period of the input signal to read the samples to be sure that nothing is lost. Depending on the prescaler divide ratio, pulses width from 0.119 microseconds to three seconds can be measured.

NOTE

A software option is provided to also generate an interrupt after the first edge.

In the example shown in **Figure 9-16**, a counter submodule is used as the time-base for a DASM configured in the input pulse width measurement mode. When the leading edge (programmed for either rising or falling edge) of the input signal occurs, the state of the time-base bus is saved in register B1. When the trailing edge occurs, the time-base bus is latched into register A, and the content of register B1 is transferred to register B2. This operation leaves register B1 free for the next leading edge to occur, as soon as on the next clock cycle. When enabled, an interrupt is provided after the trailing edge, to notify the software that pulse width measurement data is available for a new pulse. After the trailing edge, the software has one cycle time of the input signal to obtain the values for each edge. When software attention is not needed for every pulse, the interrupt can be disabled. The software can at any time read registers A and B2 coherently (using a 32-bit read instruction) to get the latest edge measurements. Since the measurement resolution is 16 bits, signals with pulse duty cycles from 0.0015% to 99.9985% can be measured. The software work is less than half that needed with a timer that requires the software to read one edge and save the value, and then wait for the second edge.

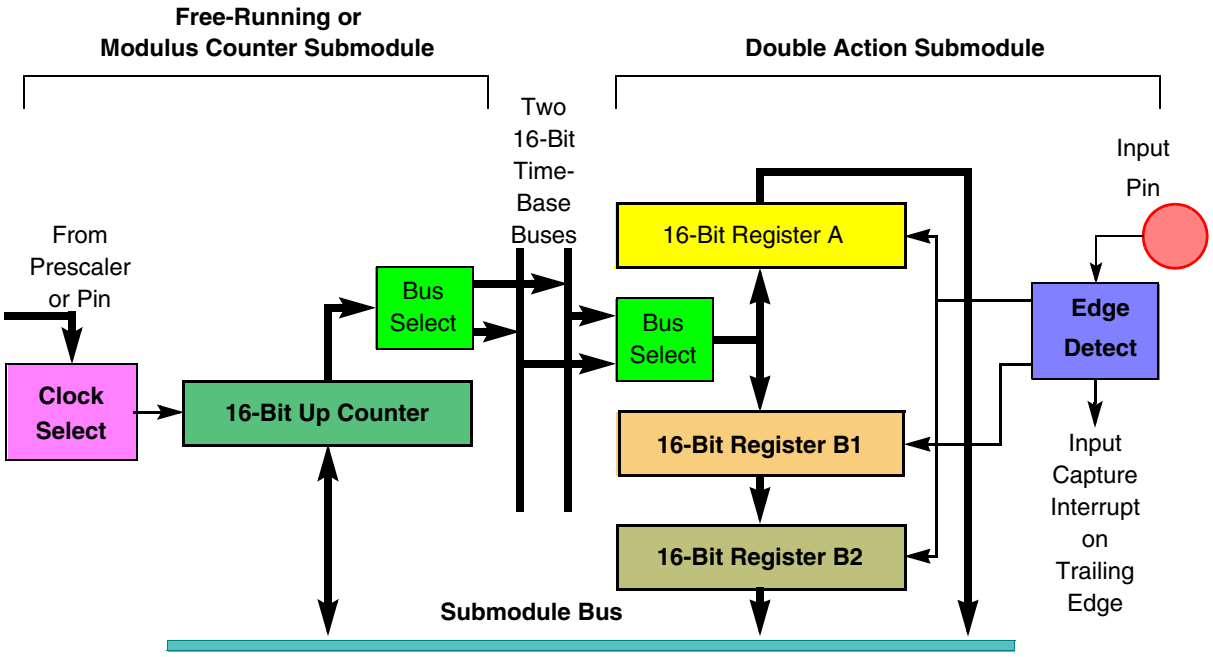


Figure 9-16 CTM9 Example — Double Capture Pulse Width Measurement

9.11.3 CTM9 Input Double Edge Period Measurement

Two samples are also available to the software from a double-action submodule for period measurement. The software can read the previous and the current edge sam-

ples and subtract them. As with pulse width measurement, the software can be sure of not missing samples by insuring that the interrupt response time is faster than the fastest input period. Alternately, when the software is just interested in the latest period measurement, one 32-bit coherent read instruction can get both the current and the previous samples. Depending on the prescaler divide ratio, period times can be measured from 0.119 microseconds to 3 seconds.

Figure 9-17 shows a counter submodule and a DASM combination as an example of period measurement. The software designates whether the rising or falling edge of the input signal is to be used for the measurements. When the edge is detected, the state of the time-base bus is stored in register A, and the content of register B1 is transferred into register B2. After register B2 is safely latched, the content of register A is transferred to register B1. This procedure gives the software coherent current and previous samples in registers A and B2 at all times. An interrupt is available for the cases where the software needs to be aware of each new sample. Note that a software option is provided to also generate an interrupt after the first edge.

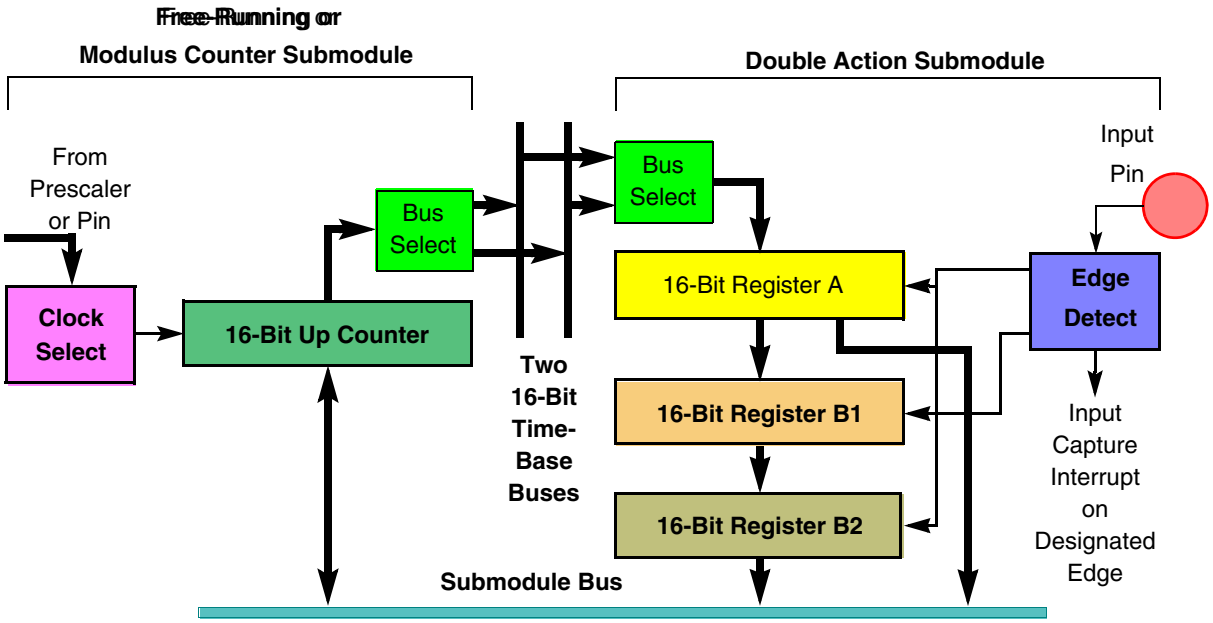


Figure 9-17 CTM9 Example — Double Capture Period Measurement

9.11.4 CTM9 Single Output Compare

To create one output edge, the software can use an SASM channel. The software provides a compare value in a register and the SASM compares that value to the incrementing value seen on one of the time-base buses. When a comparison is detected, the state of the output pin is changed.

The example shown in **Figure 9-18** uses a counter submodule with one channel of an SASM to create an output signal. The software can read the current state of the

counter submodule. That, or some other criteria, is used to determine the time-base value when the output is to change. The software thus writes the compare value into register A. In the SASM control register, the software establishes whether the output flip-flop is to toggle to the opposite state, or is to go to a high or a low level. The output compare interrupt is typically used to notify the software that the previous compare is complete and the SASM is available for a new compare value in register A.

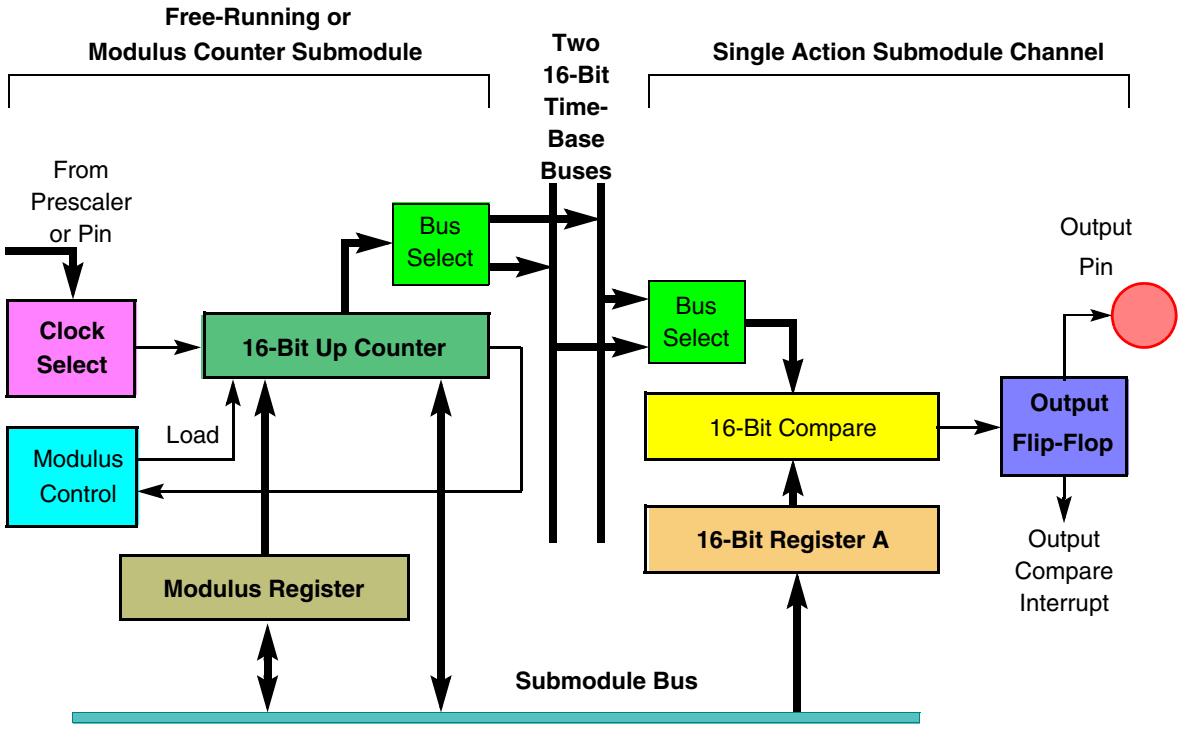


Figure 9-18 CTM9 Example — Single Edge Output Compare

9.11.5 CTM9 Double Edge Single Output Pulse Generation

Software can initialize the CTM9 to generate both the rising and the falling edge of an output pulse. With a DASM, pulses as narrow as 0.119 microseconds can be generated since software action is not needed between the edges. Pulses as long as three seconds can be generated. When an interrupt is desired, it can be selected to occur on every edge or only after the second edge.

Figure 9-19 shows how a counter submodule and a DASM can be used to generate both edges of a single output pulse. The software puts the compare value for one edge in register A and the other one in register B2. The DASM automatically creates both edges, and the pulse can be selected by software to be a high-going or a low-going. After the trailing edge, the DASM stops to await further commands from the software. Note that a single edge output can be generated by writing to only one register.

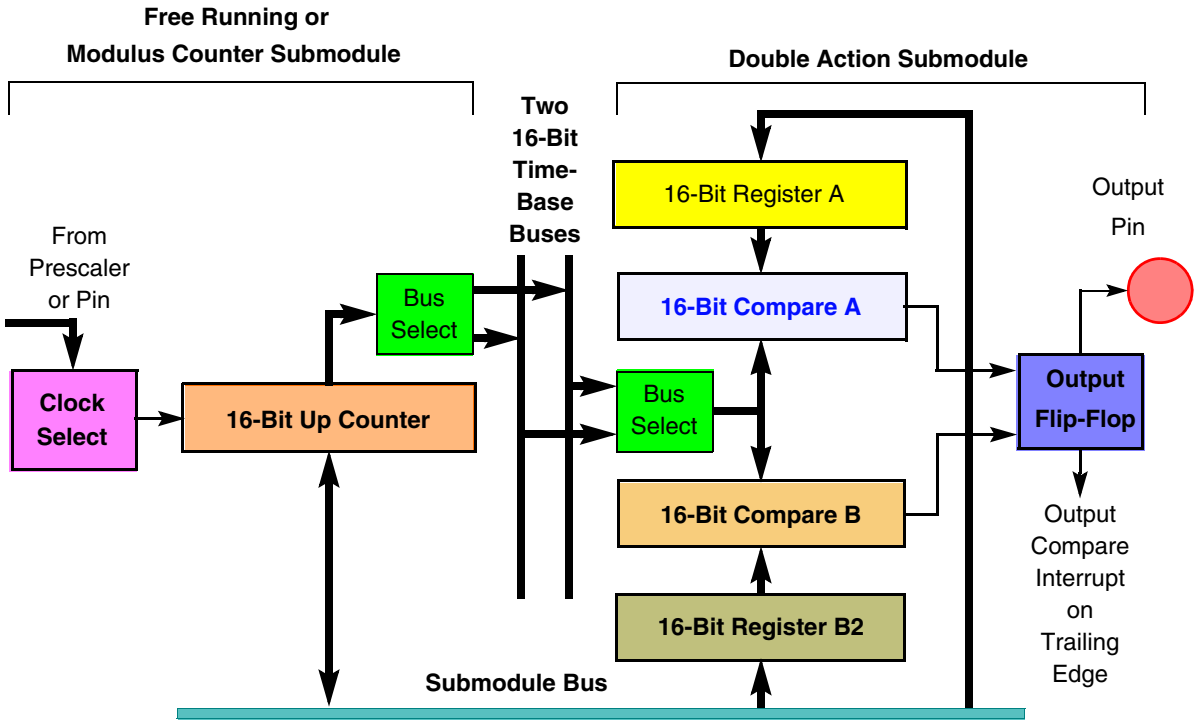


Figure 9-19 CTM9 Example — Double Edge Output Compare

9.11.6 CTM9 Output Pulse Width Modulation With DASM

Output waveforms can be generated with any duty cycle without software involvement. The software sets up a DASM with the compare times for the rising and falling edges and they are automatically repeated. The software does not need to respond to interrupts to generate continuous pulses. The period may be selected as the period of a free-running counter submodule time-base, times a binary multiplier selected in the DASM. Multiple PWM outputs can be created from multiple DASMs and share one counter submodule, provided that the periods of all of the output signals are a binary multiple of the time-base, and that the counter submodule is operating in a free-running mode. Each DASM has a software selectable “don’t care” on high-order bits of the time-base comparison so that the period of one output can be a binary multiple of another signal. Masking the time-base serves to multiply the period of the time-base by a binary number to form the period of the output waveform. The duty cycle can vary from one cycle to 64-Kbyte cycles. The frequency can range from 0.3 Hz to 62.5 KHz, though the resolution decreases at the higher frequencies to as low as 7 bits. The generation of output square wave signals is of course the special case where the high and low time are equal.

When an MCSM is used to drive the time-base, the modulus value is the period of the output PWM signal. **Figure 9-20** shows such an example. The polarity of the leading edge of an output waveform is programmable for a rising or a falling edge. The software selects the period of the output signal by programming the MCSM with a modulus

value. The leading edge compare value is written into register A by software, and the trailing edge time is written into register B1. When the leading edge value is reached, the content of register B1 is transferred to register B2, to form the next trailing edge value. Subsequent changes to the output pulse width are made by writing a new time into register B1. Updates to the pulse width are always synchronized to the leading edge of the waveform.

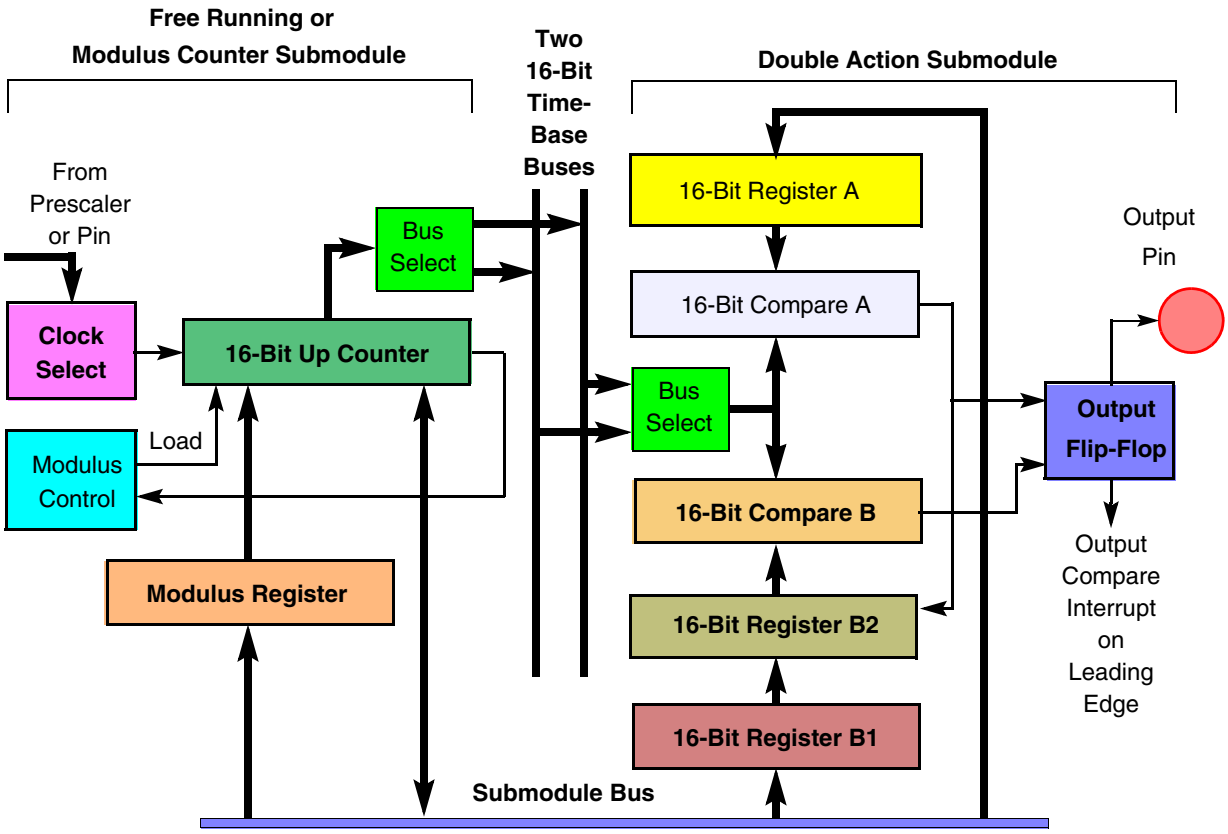


Figure 9-20 CTM9 Example — Pulse Width Modulation Output

It is typical to use the pulse width modulation mode of the DASM without interrupts, though an interrupt can be enabled to occur on the leading edge. When the output is an unchanging repetitive waveform, the DASM continuously generates the signal without any software intervention. When the software needs to change the pulse width, a new trailing edge time is written to the DASM. The output is changed on the next full pulse. When the software needs to change the output at a regular rate, such as an acceleration curve, the leading edge interrupt gives the software one period time to update the new trailing edge time.

9.11.7 CTM9 Input Pulse Accumulation

Counting the number of pulses on an input signal is another capability of the CTM9. Pulse accumulation uses either an FCSM or an MCSM. Since the counters in the

counter submodules are software readable, pulse accumulation does not require the use of an action submodule. The pulse accumulation can operate continuously, interrupting only on binary overflow of the 16-bit counter. When an MCSM is used, an interrupt can instead be created when the pulse accumulation reaches a preprogrammed value. To do that, the two's complement of the value is put in the modulus register and the interrupt occurs when the counter overflows. A similar function can be accomplished with the free-running counter submodule by writing a value into the counter.

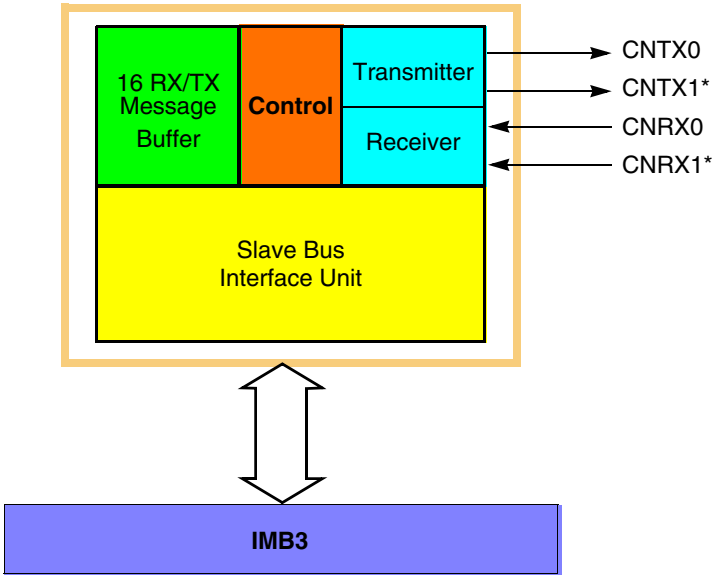
SECTION 10
CAN 2.0B CONTROLLER MODULE (TouCAN™)

10.1 Overview

MC68377 contains one CAN 2.0B controller module (TouCAN™). The TouCAN is a communication controller that implements the controller area network (CAN) protocol, an asynchronous communications protocol used in automotive and industrial control systems. It is a high speed (one Mbit/sec), short distance, priority based protocol that can run over a variety of mediums (for example, fiber optic cable or an unshielded twisted pair of wires). The TouCAN supports both the standard and extended identifier (ID) message formats specified in the CAN protocol specification, revision 2.0, part B.

The TouCAN module contains 16 message buffers, which are used for transmit and receive functions. It also contains message filters, which are used to qualify the received message IDs when comparing them to the receive buffer identifiers.

Figure 10-1 shows a block diagram of a TouCAN module.



*Note: CNTX1 and CNRX1 are not available on EXCALIBUR.

Figure 10-1 TouCAN Block Diagram

10.2 Features

The TouCAN module provides these features:

- Full implementation of CAN protocol specification, version 2.0 A/B
  - — Standard data and remote frames (up to 109 bits long)
  - — Extended data and remote frames (up to 127 bits long)
  - — Zero to eight bytes data length
  - — Programmable bit rate up to 1 Mbit/sec
- 16 RX/TX message buffers of zero to eight bytes data length
- Content-related addressing
- No read/write semaphores required
- Three programmable mask registers: global (for message buffers 0 through 13), special for message buffer 14, and special for message buffer 15
- Programmable transmit-first scheme: lowest ID or lowest buffer number
- “Time stamp”, based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Programmable I/O modes
- Maskable interrupts
- Independent of the transmission medium (external transceiver is assumed)
- Open network architecture
- Multimaster concept
- High immunity to EMI
- Short latency time for high-priority messages
- Low power sleep mode with programmable wakeup on bus activity
- Outputs have open drain drivers
- Can be used to implement recommended practices SAE J1939 and SAE J2284
- Can also be used to implement popular industrial automation standards such as DeviceNet™ and Smart Distributed System
- Motorola IMB-family modular architecture

10.3 External Pins

The TouCAN module interface to the CAN bus consists of four pins: CANTX0 and CANTX1, which transmit serial data, and CANRX0 and CANRX1, which receive serial data. **Figure 10-2** shows a typical CAN system.

NOTE

CNTX1 and CNRX1 are not available on MC68377.

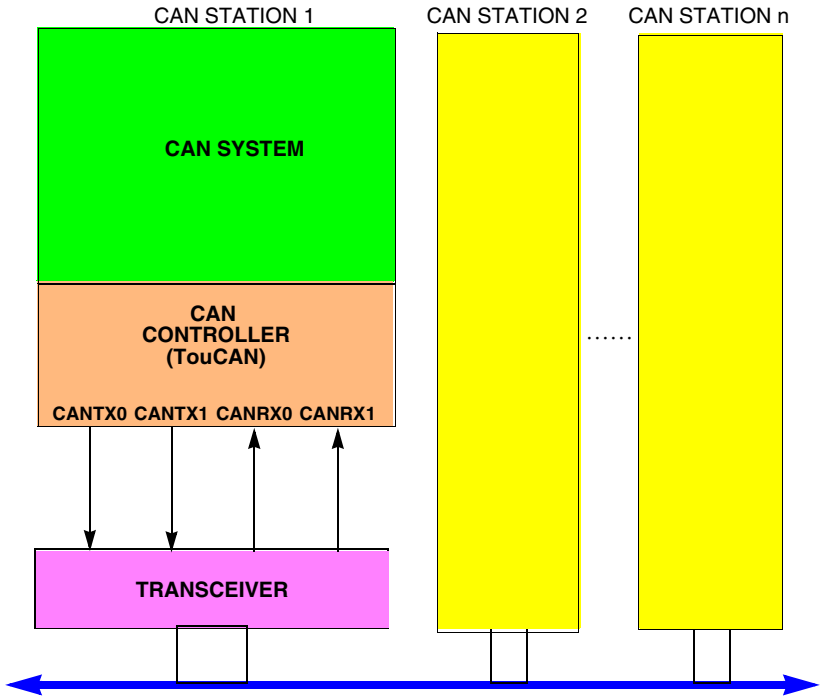


Figure 10-2 Typical CAN Network

Each CAN station is connected physically to the CAN bus through a transceiver. The transceiver provides the transmit drive, waveshaping, and receive/compare functions required for communicating on the CAN bus. It can also provide protection against damage to the TouCAN caused by a defective CAN bus or a defective CAN station.

10.4 TouCAN Architecture

The TouCAN module uses a flexible design that allows each of its 16 message buffers to be designated either a transmit (TX) buffer or a receive (RX) buffer. In addition, to reduce the CPU overhead required for message handling, each message buffer is assigned an interrupt flag bit to indicate that the transmission or reception completed successfully.

10.4.1 TX/RX Message Buffer Structure

Table 10-1 displays the extended (29 bit) ID message buffer structure. Table 10-2 displays the standard (11 bit) ID message buffer structure.

Table 10-1 Extended ID Message Buffer Structure

	158	74	30	
0x0	TIME STAMP	CODE	LENGTH	CONTROL/STATUS
0x2	ID[28-18]	SRR IDE	ID[17-15]	ID_HIGH
0x4	ID[14-0]		RTR	ID_LOW
0x6	DATA BYTE 0		DATA BYTE 1	
0x8	DATA BYTE 2		DATA BYTE 3	
0xA	DATA BYTE 4		DATA BYTE 5	
0xC	DATA BYTE 6		DATA BYTE 7	
0xE	RESERVED			

Table 10-2 Standard ID Message Buffer Structure

	158	74	30				
0x0	TIME STAMP	CODE	LENGTH				CONTROL/STATUS
0x2	ID[28:18]	RTR	0	0	0	0	ID_HIGH
0x4	16-BIT TIME STAMP						ID_LOW
0x6	DATA BYTE 0		DATA BYTE 1				
0x8	DATA BYTE 2		DATA BYTE 3				
0xA	DATA BYTE 4		DATA BYTE 5				
0xC	DATA BYTE 6		DATA BYTE 7				
0xE	RESERVED						

10.4.1.1 Common Fields for Extended and Standard Format Frames

Table 10-3 describes the message buffer fields that are common to both extended and standard identifier format frames.

Table 10-3 Common Extended/Standard Format Frames

Field	Description
Time Stamp	Contains a copy of the high byte of the free running timer, which is captured at the beginning of the identifier field of the frame on the CAN bus.
Code	Refer to Table 10-4 and Table 10-5.
RX Length	Length (in bytes) of the RX data stored in offset 0x6 through 0xD of the buffer.This field is written by the TouCAN module, copied from the DLC (data length code) field of the received frame.
TX Length	Length (in bytes) of the data to be transmitted, located in offset 0x6 through 0xD of the buffer. This field is written by the CPU and is used as the DLC field value. If RTR (remote transmission request) = 1, the frame is a remote frame and will be transmitted without data field, regardless of the value in TX length.
Data	This field can store up to eight data bytes for a frame. For RX frames, the data is stored as it is received from the bus. For TX frames, the CPU provides the data to be transmitted within the frame.
Reserved	The CPU controls access to this word entry field (16 bits).

Table 10-4 Message Buffer Codes for Receive Buffers

RX Code Before RX New Frame	Description	RX Code After RX New Frame	Comment
0000	NOT ACTIVE — message buffer is not active.	—	—
0100	EMPTY — message buffer is active and empty.	0010	—
0010	FULL — message buffer is full.	0110	If a CPU read occurs before the new frame, new receive code is 0010.
0110	OVERRUN — second frame was received into a full buffer before the CPU read the first one.		
0XY1[1]	BUSY — message buffer is now being filled with a new receive frame. This condition will be cleared within 20 cycles.	0010	An empty buffer was filled (XY was 10).
		0110	A full/overrun buffer was filled (Y was 1).

NOTES:
1. For TX message buffers, upon read, the BUSY bit should be ignored.

Table 10-5 Message Buffer Codes for Transmit Buffers

RTR	Initial TX Code	Description	Code After Successful Transmission
X	1000	Message buffer not ready for transmit.	—
0	1100	Data frame to be transmitted once, unconditionally.	1000
1	1100	Remote frame to be transmitted once, and message buffer becomes an RX message buffer for data frames.	0100
0	1010[1]	Data frame to be transmitted only as a response to a remote frame, always.	1010
0	1110	Data frame to be transmitted only once, unconditionally, and then only as a response to remote frame, always.	1010

NOTES:
1. When a matching remote request frame is detected, the code for such a message buffer is changed to be 1110.

10.4.1.2 Fields for Extended Format Frames

Table 10-6 describes the message buffer fields used only for extended identifier format frames.

Table 10-6 Extended Format Frames

Field	Description
ID[28:18]/[17:15]	Contains the 14 most significant bits of the extended identifier, located in the ID HIGH word of the message buffer.
Substitute Remote Request (SRR)	Contains a fixed recessive bit, used only in extended format. Should be set to one by the user for TX buffers. It will be stored as received on the CAN bus for RX buffers.
ID Extended (IDE)	If extended format frame is used, this field should be set to one. If zero, standard format frame should be used.
ID[14:0]	Bits [14:0] of the extended identifier, located in the ID LOW word of the message buffer.
Remote Transmission Request (RTR)	This bit is located in the least significant bit of the ID LOW word of the message buffer; 0 = Data Frame, 1 = Remote Frame.

10.4.1.3 Fields for Standard Format Frames

Table 10-7 describes the message buffer fields used only for standard identifier format frames.

Table 10-7 Standard Format Frames

Field	Description
16-Bit Time Stamp	The ID LOW word, which is not needed for standard format, is used in a standard format buffer to store the 16-bit value of the free-running timer which is captured at the beginning of the identifier field of the frame on the CAN bus.
ID[28:18]	Contains bits [28:18] of the identifier, located in the ID HIGH word of the message buffer. The four least significant bits in this register (corresponding to the IDE bit and ID[17:15] for an extended identifier message) must all be written as logic zeros to ensure proper operation of the TouCAN.
RTR	This bit is located in the ID HIGH word of the message buffer; 0 = data frame, 1 = remote frame.
RTR/SRR Bit Treatment	If the TouCAN transmits this bit as a one and receives it as a zero, an “arbitration loss” is indicated. If the TouCAN transmits this bit as a zero and is receives it as a one, a bit error is indicated. If the TouCAN transmits a value and receives a matching response, a successful bit transmission is indicated.

10.4.1.4 Serial Message Buffers

To allow double buffering of messages, the TouCAN has two shadow buffers called serial message buffers. The TouCAN uses these two buffers for buffering both received messages and messages to be transmitted. Only one serial message buffer is active at a time, and its function depends upon the operation of the TouCAN at that time. At no time does the user have access to or visibility of these two buffers.

10.4.1.5 Message Buffer Activation/Deactivation Mechanism

Each message buffer must be activated once the user configures it for the desired operation. A buffer is activated by writing the appropriate code to the control/status word for that buffer. Once the buffer is activated, it will begin participating in the normal transmit and receive processes.

A buffer is deactivated by writing the appropriate deactivation code to the control/status word for that buffer. A buffer is typically deactivated when the user desires to reconfigure the buffer (for example to change the buffer's function from RX to TX or TX to RX). The buffer should also be deactivated before changing a receive buffer's message identifier or before loading a new message to be transmitted into a transmit buffer.

For more details on activation and deactivation of message buffers and the effects on message buffer operation, refer to **10.5 TouCAN Operation**.

10.4.1.6 Message Buffer Lock/Release/Busy Mechanism

In addition to the activation/deactivation mechanism, the TouCAN also uses a lock/release/busy mechanism to ensure data coherency during the receive process. The mechanism includes a lock status for each message buffer and uses the two serial message buffers to facilitate frame transfers within the TouCAN.

Reading the control/status word of a receive message buffer triggers the lock for that buffer. While locked, a received message cannot be transferred into that buffer from one of the serial message buffers.

If a message transfer between the message buffer and a serial message buffer is in progress when the control/status word is read, the BUSY status is indicated in the code field, and the lock is not activated.

The user can release the lock on a message buffer in one of two ways. Reading the control/status word of another message buffer locks that buffer, releasing the previously locked buffer. A global release can also be performed on any locked message buffer by reading the free-running timer.

Once a lock is released, any message transfers between a serial message buffer and a message buffer that were delayed due to that buffer being locked will take place. For more details on the message buffer locking mechanism, and the effects on message buffer operation, refer to **10.5 TouCAN Operation**.

10.4.2 Receive Mask Registers

The receive mask registers are used as acceptance masks for received frame IDs. The following masks are defined:

- A global mask, used for receive buffers 0-13
- Two separate masks for buffers 14 and 15

The value of the mask registers should not be changed during normal operation. If the mask register data is changed after the masked identifier of a received message is matched to a locked message buffer, that message will be transferred into that message buffer once it is unlocked, regardless of whether that message's masked identifier still matches the receive buffer identifier. **Table 10-8** shows mask bit values.

Table 10-8 Receive Mask Register Bit Values

Mask Bit	Values
0	The corresponding incoming ID bit is “don’t care”.
1	The corresponding ID bit is checked against the incoming ID bit to see if a match exists.

Table 10-9 shows mask examples for normal and extended messages. Refer to 10.8 Programmer’s Model for more information on RX mask registers.

Table 10-9 Mask Examples for Normal/Extended Messages

Message Buffer (MB) /Mask	Base ID ID[28:18]	IDE	Extended ID ID[17:0]	Match
MB2	1 1 1 1 1 1 1 1 1 0 0 0	0	—	—
MB3	1 1 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	—
MB4	0 0 0 0 0 0 0 1 1 1 1 1	0	—	—
MB5	0 0 0 0 0 0 0 1 1 1 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	—
MB14	1 1 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	—
RX Global Mask	1 1 1 1 1 1 1 1 1 1 1 0		1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1	—
RX Message In	1 1 1 1 1 1 1 1 1 0 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	3[1]
	1 1 1 1 1 1 1 1 1 0 0 1	0	—	2[2]
	1 1 1 1 1 1 1 1 1 0 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0	—[3]
	0 1 1 1 1 1 1 1 1 0 0 0	0	—	—[4]
	0 1 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	—[5]
RX 14 Mask	0 1 1 1 1 1 1 1 1 1 1 1		1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0	—
RX Message In	1 0 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	—[6]
	0 1 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	14[7]

NOTES:
1. Match for extended format (MB3).
2. Match for standard format (MB2).
3. No match for MB3 because of ID0.
4. No match for MB2 because of ID28.
5. No match for MB3 because of ID28, match for MB14.
6. No match for MB14 because of ID27.
7. Match for MB14.

10.4.3 Bit Timing

The TouCAN module uses three 8-bit registers to set up the bit timing parameters required by the CAN protocol. Control registers 1 and 2 (CANCTRL1, CANCTRL2) contain the PROPSEG, PSEG1, PSEG2, and the RJW fields which allow the user to configure the bit timing parameters. The prescaler divide register (PRESDIV) allows the user to select the ratio used to derive the S-clock from the system clock. The time quanta clock operates at the S-clock frequency. Table 10-10 provides examples of system clock, CAN bit rate, and S-clock bit timing parameters. Refer to 10.8 Programmer’s Model for more information on the bit timing registers.

Table 10-10 Example System Clock, CAN Bit Rate and S-Clock Frequencies

System Clock Frequency (MHz)	CAN Bit-Rate (MHz)	Possible S-Clock Frequency (MHz)	Possible Number of Time Quanta/Bit	PRESDIV Value + 1
25	1	25	25	1
20	1	10, 20	10, 20	2, 1
16	1	8, 16	8, 16	2, 1
25	0.125	1, 1.25, 2.5	8,10, 20	25, 20,10
20	0.125	1, 2, 2.5	8, 16, 20	20, 10, 8
16	0.125	1, 2	8,16	16, 8

10.4.3.1 Configuring the TouCAN Bit Timing

The following considerations must be observed when programming bit timing functions.

- If the programmed PRESDIV value results in a single system clock per one time quantum, then the PSEG2 field in CANCTRL2 register must not be programmed to zero.
- If the programmed PRESDIV value results in a single system clock per one time quantum, then the information processing time (IPT) equals three time quanta; otherwise it equals two time quanta. If PSEG2 equals two, then the TouCAN transmits one time quantum late relative to the scheduled sync segment.
- If the prescaler and bit timing control fields are programmed to values that result in fewer than 10 system clock periods per CAN bit time and the CAN bus loading is 100%, then anytime the rising edge of a start-of-frame (SOF) symbol transmitted by another node occurs during the third bit of the intermission between messages, the TouCAN may not be able to prepare a message buffer for transmission in time to begin its own transmission and arbitrate against the message which transmitted the early SOF.
- The TouCAN bit time must be programmed to be greater than or equal to nine system clocks, or correct operation is not guaranteed.

10.4.4 Error Counters

The TouCAN has two error counters: the transmit (TX) error counter and the receive (RX) error counter. Refer to **10.8 Programmer's Model** for more information on error counters.The rules for increasing and decreasing these counters are described in the CAN protocol, and are fully implemented in the TouCAN. Each counter has the following features:

- 8-bit up/down counter
- Increment by eight (RX error counter also increments by one)
- Decrement by one
- Avoid decrement when equal to zero
- RX error counter reset to a value between 119 and 127 inclusive, when the TouCAN transitions from error passive to error active
- Following reset, both counters reset to zero

- Detect values for error passive, bus off and error active transitions
- Cascade usage of TX error counter with an additional internal counter to detect the 128 occurrences of 11 consecutive recessive bits necessary to transition from bus off into error active.

Both counters are read only (except in test/freeze/halt modes).

The TouCAN responds to any bus state as described in the CAN protocol, transmitting an error active or error passive flag, delaying its transmission start time (error passive) and avoiding any influence on the bus when in the bus off state. The following are the basic rules for TouCAN bus state transitions:

- If the value of the TX error counter or RX error counter increments to a value greater than or equal to 128, the fault confinement state (FCS[1:0]) field in the error status register is updated to reflect an error passive state.
- If the TouCAN is in an error passive state, and either the TX error counter or RX error counter decrements to a value less than or equal to 127 while the other error counter already satisfies this condition, the FCS[1:0] field in the error status register is updated to reflect an error active state.
- If the value of the TX error counter increases to a value greater than 255, the FCS[1:0] field in the error status register is updated to reflect a bus off state, and an interrupt may be issued. The value of the TX error counter is reset to zero.
- If the TouCAN is in the bus off state, the TX error counter and an additional internal counter are cascaded to count 128 occurrences of 11 consecutive recessive bits on the bus. To do this, the TX error counter is first reset to zero, and then the internal counter begins counting consecutive recessive bits. Each time the internal counter counts 11 consecutive recessive bits, the TX error counter is incremented by one and the internal counter is reset to zero. When the TX error counter reaches the value of 128, the FCS[1:0] field in the error status register is updated to be error active, and both error counters are reset to zero. Any time a dominant bit is detected following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero but does not affect the TX error counter value.
- If only one node is operating in a system, the TX error counter is incremented with each message it attempts to transmit, due to the resulting acknowledgment errors. However, acknowledgment errors never cause the TouCAN to change from the error passive state to the bus off state.
- If the RX error counter increments to a value greater than 127, it stops incrementing, even if more errors are detected while being a receiver. After the next successful message reception, the counter is reset to a value between 119 and 127, to enable a return to the error active state.

10.4.5 Time Stamp

The value of the free-running 16-bit timer is sampled at the beginning of the identifier field on the CAN bus. For a message being received, the time stamp is stored in the time stamp entry of the receive message buffer at the time the message is written into that buffer. For a message being transmitted, the time stamp entry is written into the transmit message buffer once the transmission has completed successfully.

The free-running timer can optionally be reset upon the reception of a frame into message buffer 0. This feature allows network time synchronization to be performed.

10.5 TouCAN Operation

The basic operation of the TouCAN can be divided into three areas:

- Reset and initialization of the module
- Transmit message handling
- Receive message handling

Example sequences for performing each of these processes is given in the following paragraphs.

10.5.1 TouCAN Reset

The TouCAN can be reset in two ways:

- Hard reset, using one of the IMB3 reset lines.
- Soft reset, using the SOFTRST bit in the module configuration register.

Following the negation of reset, the TouCAN is not synchronized with the CAN bus, and the HALT, FRZ, and FRZACK bits in the module configuration register are set. In this state, the TouCAN does not initiate frame transmissions or receive any frames from the CAN bus. The contents of the message buffers are not changed following reset.

Any configuration change or initialization requires that the TouCAN be frozen by either the assertion of the HALT bit in the module configuration register or by reset.

10.5.2 TouCAN Initialization

Initialization of the TouCAN includes the initial configuration of the message buffers and configuration of the CAN communication parameters following a reset, as well as any reconfiguration which may be required during operation. The following is a general initialization sequence for the TouCAN:

1. Initialize all operation modes
   a. Initialize the transmit and receive pin modes in control register 0 (CANCTRL0).
   b. Initialize the bit timing parameters PROPSEG, PSEGS1, PSEG2, and RJW in control registers 1 and 2 (CANCTRL[1:2]).
   c. Select the S-clock rate by programming the PRESDIV register.
   d. Select the internal arbitration mode (LBUF bit in CANCTRL1).
2. Initialize message buffers
   a. The control/status word of all message buffers must be written either as an active or inactive message buffer.
   b. All other entries in each message buffer should be initialized as required.

3. Initialize mask registers for acceptance mask as needed
4. Initialize TouCAN interrupt handler

   a. Initialize the interrupt configuration register (CANICR) with a specific request level
   b. Set the required mask bits in the IMASK register (for all message buffer interrupts), in CANCTRL0 (for bus off and error interrupts), and in CANMCR for the WAKE interrupt.

5. Negate the HALT bit in the module configuration register

   a. At this point, the TouCAN attempts to synchronize with the CAN bus.

NOTE

In both the transmit and receive processes, the first action in preparing a message buffer must be to deactivate the buffer by setting its code field to the proper value. This step is mandatory to ensure data coherency.

10.5.3 Transmit Process

The transmit process includes preparation of a message buffer for transmission, as well as the internal steps performed by the TouCAN to decide which message to transmit. For the user, this involves loading the message and ID to be transmitted into a message buffer and then activating that buffer as an active transmit buffer. Once this is done, the TouCAN performs all additional steps necessary to transmit the message onto the CAN bus.

The user should prepare or change a message buffer for transmission by executing the following steps.

1. Write the control/status word to hold the transmit buffer inactive (code = 0b1000)
2. Write the ID_HIGH and ID_LOW words
3. Write the data bytes
4. Write the control/status word (active TX code, TX length)

NOTE

Steps 1 and 4 are mandatory to ensure data coherency.

Once an active transmit code is written to a transmit message buffer, that buffer begins participating in an internal arbitration process as soon as the receiver senses that the CAN bus is free, or at the inter-frame space. If there are multiple messages awaiting transmission, this internal arbitration process selects the message buffer from which the next frame is transmitted.

When this process is over and a message buffer is selected for transmission, the frame from that message buffer is transferred to the serial message buffer for transmission.

The TouCAN transmit s no more than eight data bytes, even if the transmit length contains a value greater than eight.

At the end of a successful transmission, the value of the free-running timer (which was captured at the beginning of the identifier field on the CAN bus), is written into the time stamp field in the message buffer. The code field in the control/status word of the message buffer is updated and a status flag is set in the IFLAG register.

10.5.3.1 Transmit Message Buffer Deactivation

Any write access to the control/status word of a transmit message buffer during the process of selecting a message buffer for transmission immediately deactivates that message buffer, removing it from the transmission process.

If the user deactivates a transmit message buffer while a message is being transferred from it to a serial message buffer, the message is not transmitted.

If the user deactivates the transmit message buffer after the message is transferred to the serial message buffer, the message is transmitted, but no interrupt is requested, and the transmit code is not updated.

If a message buffer containing the lowest ID is deactivated while that message is undergoing the internal arbitration process to determine which message should be sent, then that message may not be transmitted.

10.5.3.2 Reception of Transmitted Frames

The TouCAN receives a frame it has transmitted if an empty message buffer with a matching identifier exists.

10.5.4 Receive Process

During the receive process, the following events occur:

- The user configures the message buffers for reception.
- The TouCAN transfers received messages from the serial message buffers to the receive message buffers with matching IDs.
- The user retrieves these messages.

The user should prepare or change a message buffer for frame reception by executing the following steps.

1. Write the control/status word to hold the receive buffer inactive (code = 0b0000).
2. Write the ID_HIGH and ID_LOW words.
3. Write the control/status word to mark the receive message buffer as active and empty.

NOTE

Steps 1 and 3 are mandatory for data coherency.

Once these steps are performed, the message buffer functions as an active receive buffer and participates in the internal matching process, which takes place every time the TouCAN receives an error-free frame. In this process, all active receive buffers compare their ID value to the newly received one. If a match is detected, the following actions occur:

1. The frame is transferred to the first (lowest entry) matching receive message buffer.
2. The value of the free-running timer (captured at the beginning of the identifier field on the CAN bus) is written into the time stamp field in the message buffer.
3. The ID field, data field, and RX length field are stored.
4. The code field is updated.
5. The status flag is set in the IFLAG register.

The user should read a received frame from its message buffer in the following order:

1. Control/status word (mandatory, as it activates the internal lock for this buffer)
2. ID (optional, since it is needed only if a mask was used)
3. Data field word(s)
4. Free-running timer (optional, as it releases the internal lock)

If the free running timer is not read, that message buffer remains locked until the read process starts for another message buffer. Only a single message buffer is locked at a time. When a received message is read, the only mandatory read operation is that of the control/status word. This ensures data coherency.

If the BUSY bit is set in the message buffer code, the CPU should defer accessing that buffer until this bit is negated. Refer to **Table 10-4**.

NOTE

The user should check the status of a message buffer by reading the status flag in the IFLAG register and not by reading the control/status word code field for that message buffer. This prevents the buffer from being locked inadvertently.

Because the received identifier field is always stored in the matching receive message buffer, the contents of the identifier field in a receive message buffer may change if one or more of the ID bits are masked.

10.5.4.1 Receive Message Buffer Deactivation

Any write access to the control/status word of a receive message buffer during the process of selecting a message buffer for reception immediately deactivates that message buffer, removing it from the reception process.

If a receive message buffer is deactivated while a message is being transferred into it, the transfer is halted and no interrupt is requested. If this occurs, that receive message buffer may contain mixed data from two different frames.

Data must never be written into a receive message buffer. If this occurs while a message is being transferred from a serial message buffer, the control/status word will reflect a full or overrun condition, but no interrupt is requested.

10.5.4.2 Locking and Releasing Message Buffers

The lock/release/busy mechanism is designed to guarantee data coherency during the receive process. The following examples demonstrate how the the lock/release/busy mechanism affects TouCAN operation.

1. Reading a control/status word of a message buffer triggers a lock for that message buffer. A new received message frame which matches the message buffer cannot be written into this message buffer while it is locked.
2. To release a locked message buffer, the CPU either locks another message buffer by reading its control/status word or globally releases any locked message buffer by reading the free-running timer.
3. If a receive frame with a matching ID is received during the time the message buffer is locked, the receive frame is not immediately transferred into that message buffer, but remains in the serial message buffer. There is no indication when this occurs.
4. When a locked message buffer is released, if a frame with a matching identifier exists within the serial message buffer, then this frame is transferred to the matching message buffer.
5. If two or more receive frames with matching IDs are received while a message buffer with a matching ID is locked, the last received frame with that ID is kept within the serial message buffer, while all preceding ones are lost. There is no indication when this occurs.
6. If the user reads the control/status word of a receive message buffer while a frame is being transferred from a serial message buffer, the BUSY code is indicated. The user should wait until this code is cleared before continuing to read from the message buffer to ensure data coherency. In this situation, the read of the control/status word does not lock the message buffer.

Polling the control/status word of a receive message buffer can lock it, preventing a message from being transferred into that buffer. If the control/status word of a receive message buffer is read, it should be followed by a read of the control/status word of another buffer, or by a read of the free-running timer, to ensure that the locked buffer is unlocked.

10.5.5 Remote Frames

The remote frame is a message frame that is transmitted to request a data frame. The TouCAN can be configured to transmit a data frame automatically in response to a remote frame or to transmit a remote frame and then wait for the responding data frame to be received.

To transmit a remote frame, the user initializes a message buffer as a transmit message buffer with the RTR bit set to one. Once this remote frame is transmitted successfully, the transmit message buffer automatically becomes a receive message buffer with the same ID as the remote frame that was transmitted.

When the TouCAN receives a remote frame, it compares the remote frame ID to the IDs of all transmit message buffers programmed with a code of 1010. If there is an exact matching ID, the data frame in that message buffer is transmitted. If the RTR bit in the matching transmit message buffer is set, the TouCAN transmits a remote frame as a response.

A received remote frame is not stored in a receive message buffer. It is only used to trigger the automatic transmission of a frame in response. The mask registers are not used in remote frame ID matching. All ID bits (except RTR) of the incoming received frame must match for the remote frame to trigger a response transmission.

10.5.6 Overload Frames

The TouCAN does not initiate overload frame transmissions unless it detects the following conditions on the CAN bus:

- A dominant bit in the first or second bit of intermission.
- A dominant bit in the seventh (last) bit of the end-of-frame (EOF) field in receive frames
- A dominant bit in the eighth (last) bit of the error frame delimiter or overload frame delimiter

10.6 Special Operating Modes

The TouCAN module has three special operating modes:

- Debug mode
- Low-power stop mode
- Auto-power save mode

10.6.1 Debug Mode

Debug mode is entered when the FRZ1 bit in CANMCR is set and one of the following events occurs:

- The HALT bit in the CANMCR is set; or
- The IMB3 FREEZE line is asserted

Once entry into debug mode is requested, the TouCAN waits until an intermission or idle condition exists on the CAN bus, or until the TouCAN enters the error passive or

bus off state. Once one of these conditions exists, the TouCAN waits for the completion of all internal activity. Once this happens, the following events occur:

- The TouCAN stops transmitting or receiving frames.
- The prescaler is disabled, thus halting all CAN bus communication.
- The TouCAN ignores its RX pins and drives its TX pins as recessive. The TouCAN loses synchronization with the CAN bus and the NOTRDY and FRZACK bits in CANMCR are set.
- The CPU is allowed to read and write the error counter registers.

After engaging one of the mechanisms to place the TouCAN in debug mode, the user must wait for the FRZACK bit to be set before accessing any other registers in the TouCAN; otherwise unpredictable operation may occur.

To exit debug mode, the IMB FREEZE line must be negated or the HALT bit in CANMCR must be cleared.

Once debug mode is exited, the TouCAN resynchronizes with the CAN bus by waiting for 11 consecutive recessive bits before beginning to participate in CAN bus communication.

10.6.2 Low-Power Stop Mode

Before entering low-power stop mode, the TouCAN waits for the CAN bus to be in an idle state, or for the third bit of intermission to be recessive. The TouCAN then waits for the completion of all internal activity (except in the CAN bus interface) to be complete. Then the following events occur:

- The TouCAN shuts down its clocks, stopping most internal circuits, thus achieving maximum power savings.
- The bus interface unit continues to operate, allowing the CPU to access the module configuration register.
- The TouCAN ignores its RX pins and drives its TX pins as recessive.
- The TouCAN loses synchronization with the CAN bus, and the STOPACK and NOTRDY bits in the module configuration register are set.

To exit low-power stop mode:

- Reset the TouCAN either by asserting one of the IMB3 reset lines or by asserting the SOFTRST bit CANMCR.
- Clear the STOP bit in CANMCR.
- The TouCAN module can optionally exit low-power stop mode via the self-wake mechanism. If the SELFWAKE bit in CANMCR was set at the time the TouCAN entered stop mode, then upon detection of a recessive to dominant transition on the CAN bus, the TouCAN clears the STOP bit in CANMCR and its clocks begin running.

When the TouCAN is in low-power stop mode, a recessive to dominant transition on the CAN bus causes the WAKEINT bit in the error and status register (ESTAT) to be set. This event generates an interrupt if the WAKEMSK bit in CANMCR is set.

Consider the following notes regarding low-power stop mode:

- When the self-wake mechanism is activated, the TouCAN tries to receive the frame that woke it up. (It assumes that the dominant bit detected is a start-of-frame bit.) It will not arbitrate for the CAN bus at this time.
- If the STOP bit is set while the TouCAN is in the bus-off state, then the TouCAN enters low-power stop mode and stops counting recessive bit times. The count continues when STOP is cleared.
- To place the TouCAN in low-power stop mode with the self-wake mechanism engaged, write to CANMCR with both STOP and SELFWAKE set, and then wait for the TouCAN to set the STOPACK bit.
- To take the TouCAN out of low-power stop mode when the self-wake mechanism is enabled, write to CANMCR with both STOP and SELFWAKE clear, and then wait for the TouCAN to clear the STOPACK bit.
- The SELFWAKE bit should not be set after the TouCAN has already entered low-power stop mode.
- If both STOP and SELFWAKE are set and a recessive to dominant edge immediately occurs on the CAN bus, the TouCAN may never set the STOPACK bit, and the STOP bit will be cleared.
- To prevent old frames from being sent when the TouCAN awakes from low-power stop mode via the self-wake mechanism, disable all transmit sources, including transmit buffers configured for remote request responses, before placing the TouCAN in low-power stop mode.
- If the TouCAN is in debug mode when the STOP bit is set, the TouCAN assumes that debug mode should be exited. As a result, it tries to synchronize with the CAN bus, and only then does it await the conditions required for entry into low-power stop mode.
- Unlike other modules, the TouCAN does not come out of reset in low-power stop mode. The basic TouCAN initialization procedure should be executed before placing the module in low-power stop mode. (Refer to **10.5.2 TouCAN Initialization**.)
- If the TouCAN is in low-power stop mode with the self-wake mechanism engaged and is operating with a single system clock per time quantum, there can be extreme cases in which TouCAN wake-up on recessive to dominant edge may not conform to the CAN protocol. TouCAN synchronization is shifted one time quantum from the wake-up event. This shift lasts until the next recessive to dominant edge, which resynchronizes the TouCAN to be in conformance with the CAN protocol. The same holds true when the TouCAN is in auto-power save mode and awakens on a recessive to dominant edge.

10.6.3 Auto-Power Save Mode

Auto power save mode enables normal operation with optimized power savings. Once the auto power save (APS) bit in CANMCR is set, the TouCAN looks for a set of conditions in which there is no need for the clocks to be running. If these conditions are met, the TouCAN stops its clocks, thus saving power. The following conditions activate auto-power save mode.

- No Rx/Tx frame in progress.
- No transfer of Rx/Tx frames to and from a serial message buffer, and no TX frame

awaiting transmission in any message buffer.
- No CPU access to the TouCAN module.
- The TouCAN is not in debug mode, low-power stop mode, or the bus off state.

While its clocks are stopped, if the TouCAN senses that any one of the aforementioned conditions is no longer true, it restarts its clocks. The TouCAN then continues to monitor these conditions and stops or restarts its clocks accordingly.

10.7 Interrupts

The TouCAN is capable of generating one interrupt level on the IMB3. This level is programmed into the priority level bits in the interrupt configuration register (CANICR). This value determines which interrupt signal is driven onto the bus when an interrupt is requested.

When an interrupt is requested, the CPU32 initiates an IACK cycle. The TouCAN decodes the IACK cycle and compares the CPU32 recognized level to the level that it is currently requesting. If a match occurs, then arbitration begins. If the TouCAN wins arbitration, it generates a uniquely encoded interrupt vector that indicates which event is requesting service. This encoding scheme is as follows:

- The higher-order bits of the interrupt vector come from the IVBA[2:0] field in CANICR.
- The low-order five bits are an encoded value that indicate which of the 19 TouCAN interrupt sources is requesting service.

Figure 10-3 shows a block diagram of the interrupt hardware.

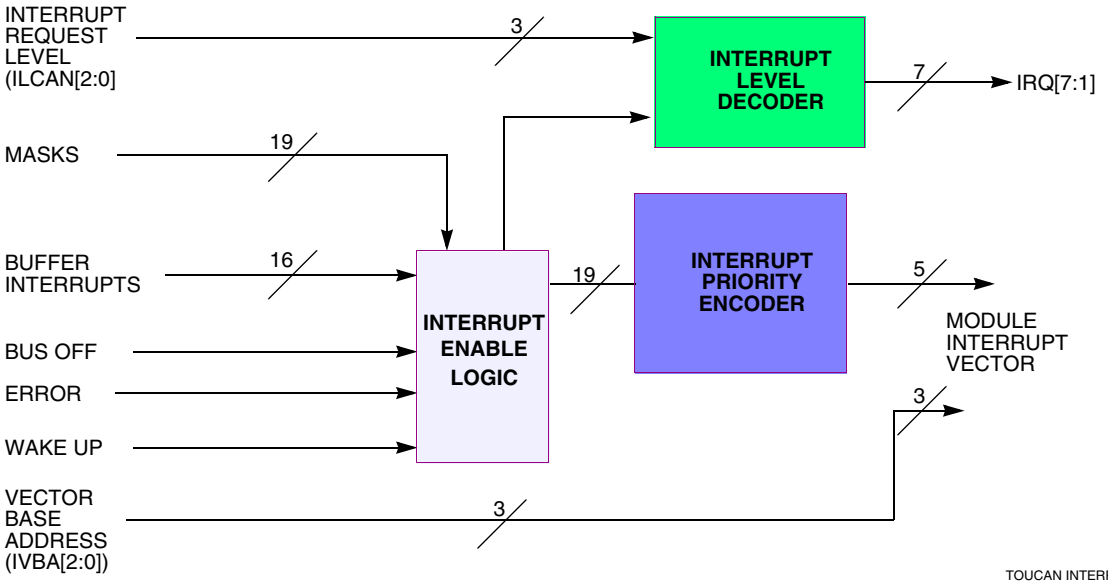


Figure 10-3 TouCAN Interrupt Vector Generation

Each one of the 16 message buffers can be an interrupt source, if its corresponding IMASK bit is set. There is no distinction between transmit and receive interrupts for a particular buffer. Each of the buffers is assigned a bit in the IFLAG register. An IFLAG bit is set when the corresponding buffer completes a successful transmission/reception. An IFLAG bit is cleared when the CPU32 reads IFLAG while the associated bit is set, and then writes it back as zero (and no new event of the same type occurs between the read and the write actions).

The other three interrupt sources (bus off, error and wake up) act in the same way, and have flag bits located in the error and status register (ESTAT). The bus off and error interrupt mask bits (BOFFMSK and ERRMSK) are located in CANCTRL0, and the wake up interrupt mask bit (WAKEMSK) is located in the module configuration register. **Table 10-11** shows TouCAN interrupt priorities and their corresponding vector addresses.

Table 10-11 Interrupt Sources and Vector Addresses

Interrupt Source	Vector Number
Buffer 0	0bXXX00000 (Highest priority)
Buffer 1	0bXXX00001
Buffer 2	0bXXX00010
Buffer 3	0bXXX00011
Buffer 4	0bXXX00100
Buffer 5	0bXXX00101
Buffer 6	0bXXX00110
Buffer 7	0bXXX00111
Buffer 8	0bXXX01000
Buffer 9	0bXXX01001
Buffer 10	0bXXX01010
Buffer 11	0bXXX01011
Buffer 12	0bXXX01100
Buffer 13	0bXXX01101
Buffer 14	0bXXX01110
Buffer 15	0bXXX01111
Bus off	0bXXX10000
Error	0bXXX10001
Wake-up	0bXXX10010 (Lowest priority)

10.8 Programmer's Model

Table 10-12 shows the TouCAN address map.

The column labeled "Access" indicates the privilege level at which the CPU must be operating to access the register. A designation of "S" indicates that supervisor mode is required. A designation of "S/U" indicates that the register can be programmed for either supervisor mode access or unrestricted access.

The address space for each TouCAN module is split, with 128 bytes starting at the base address, and an extra 256 bytes starting at the base address +128. The upper 256 are fully used for the message buffer structures. Of the lower 128 bytes, some are not used. Registers with bits marked as “reserved” should always be written as logic 0.

Typically, the TouCAN control registers are programmed during system initialization, before the TouCAN becomes synchronized with the CAN bus. The configuration registers can be changed after synchronization by halting the TouCAN module. This is done by setting the HALT bit in the TouCAN module configuration register (CANMCR). The TouCAN responds by asserting the CANMCR NOTRDY bit. Additionally, the control registers can be modified while the MCU is in background debug mode.

NOTE

The TouCAN has no hard-wired protection against invalid bit/field programming within its registers. Specifically, no protection is provided if the programming does not meet CAN protocol requirements.

Table 10-12 TouCAN Register Map

Access	Address[1]	15 8	7 0
S	0xYF FA80	TouCAN Module Configuration Register (TCNMCR) See **Table 10-13** for bit descriptions.	
S	0xYF FA82	TouCAN Test Register (TTR)	
S	0xYF FA84	TouCAN Interrupt Configuration Register (CANICR)	
S/U	0xYF FA86	Control Register 0 (CANCTRL0) See **Table 10-15** and **Table 10-18** for bit descriptions.	Control Register 1 (CANCTRL1)
S/U	0xYF FA88	Control and Prescaler Divider Register (PRESDIV) See **Table 10-19** and **Table 10-20** for bit descriptions.	Control Register 2 (CTRL2)
S/U	0xYF FA8A	Free-Running Timer Register (TIMER) See **Table 10-21** for bit descriptions.	
—	—	Reserved	
S/U	0xYF FA90	Receive Global Mask – High (RXGMASKHI) See **Table 10-22** for bit descriptions.	
S/U	0xYF FA92	Receive Global Mask – Low (RXGMASKLO) See **Table 10-22** for bit descriptions.	
S/U	0xYF FA94	Receive Buffer 14 Mask – High (RX14MASKHI) See **10.8.9 Receive Buffer 14 Mask Registers** for bit descriptions.	
S/U	0xYF FA96	Receive Buffer 14 Mask – Low (RX14MASKLO) See **10.8.9 Receive Buffer 14 Mask Registers** for bit descriptions.	
S/U	0xYF FA98	Receive Buffer 15 FMask – High (RX15MASKHI) See **10.8.10 Receive Buffer 15 Mask Registers** for bit descriptions.	
S/U	0xYF FA9A	Receive Buffer 15 Mask – Low (RX15MASKLO) See **10.8.10 Receive Buffer 15 Mask Registers** for bit descriptions.	
—	—	Reserved	
S/U	0xYF FAA0	Error and Status Register (ESTAT) See **Table 10-23** for bit descriptions.	

Table 10-12 TouCAN Register Map (Continued)

Access	Address[1]	15 8	7 0
S/U	0xYF FAA2	Interrupt Masks (IMASK) See Table 10-26 for bit descriptions.	
S/U	0xYF FAA4	Interrupt Flags (IFLAG) See Table 10-27 for bit descriptions.	
S/U	0xYF FAA6	Receive Error Counter (RXECTR) See Table 10-28 for bit descriptions.	Transmit Error Counter (TXECTR)

NOTES:
1. Y = M111, where M is the logic state of the module mapping (MM) bit in the SCIM2E configuration register (SCIMMCR).

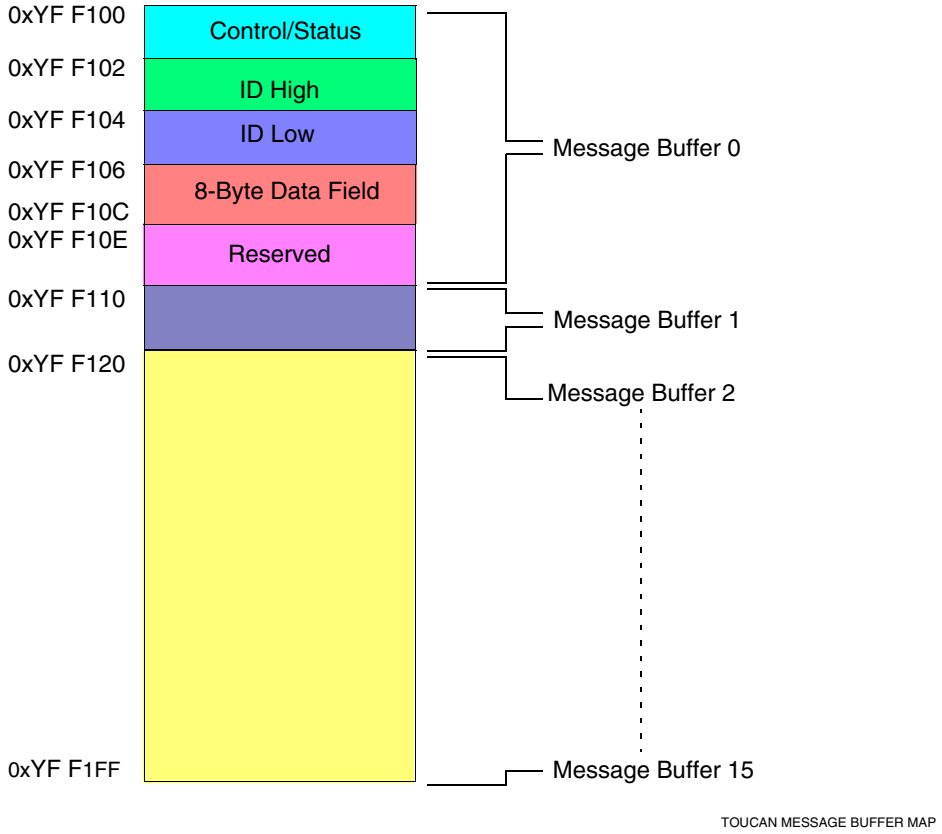


Figure 10-4 TouCAN Message Buffer Memory Map

10.8.1 TouCAN Module Configuration Register

TCNMCR — TouCAN Module Configuration Register **0xYF FA80**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
STOP	FRZ	NOT USED	HALT	NOT RDY	WAKE MSK	SOFT RST	FRZ ACK	SUPV	SELF WAKE	APS	STOP ACK	IARB[3:0]			
RESET:															
0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0

Table 10-13 TCNMCR Bit Settings

Bit(s)	Name	Description
15	STOP	Low-power stop mode enable. The STOP bit may only be set by the CPU. It may be cleared either by the CPU or by the TouCAN, if the SELFWAKE bit is set. 0 = Enable TouCAN clocks 1 = Disable TouCAN clocks
14	FRZ	FREEZE assertion response. When FRZ = 1, the TouCAN can enter debug mode when the IMB3 FREEZE line is asserted or the HALT bit is set. Clearing this bit field causes the TouCAN to exit debug mode. Refer to **10.6.1 Debug Mode** for more information. 0 = TouCAN ignores the IMB3 FREEZE signal and the HALT bit in the module configuration register. 1 = TouCAN module enabled to enter debug mode.
13	—	Reserved
12	HALT	Halt TouCAN S-Clock. Setting the HALT bit has the same effect as assertion of the IMB3 FREEZE signal on the TouCAN without requiring that FREEZE be asserted. This bit is set to one after reset. It should be cleared after initializing the message buffers and control registers. TouCAN message buffer receive and transmit functions are inactive until this bit is cleared. When HALT is set, write access to certain registers and bits that are normally read-only is allowed. 0 = The TouCAN operates normally 1 = TouCAN enters debug mode if FRZ = 1
11	NOTRDY	TouCAN not ready. This bit indicates that the TouCAN is either in low-power stop mode or debug mode. This bit is read-only and is set only when the TouCAN enters low-power stop mode or debug mode. It is cleared once the TouCAN exits either mode, either by synchronization to the CAN bus or by the self-wake mechanism. 0 = TouCAN has exited low-power stop mode or debug mode. 1 = TouCAN is in low-power stop mode or debug mode.
10	WAKEMSK	Wakeup interrupt mask. The WAKEMSK bit enables wake-up interrupt requests. 0 = Wake up interrupt is disabled. 1 = Wake up interrupt is enabled.
9	SOFTRST	Soft reset. When this bit is asserted, the TouCAN resets its internal state machines (sequencer, error counters, error flags, and timer) and the host interface registers (CANMCR, CANICR, CANTCR, IMASK, and IFLAG). The configuration registers that control the interface with the CAN bus are not changed (CANCTRL[0:2] and PRESDIV). Message buffers and receive message masks are also not changed. This allows SOFTRST to be used as a debug feature while the system is running. Setting SOFTRST also clears the STOP bit in CANMCR. After setting SOFTRST, allow one complete bus cycle to elapse for the internal TouCAN circuitry to completely reset before executing another access to CANMCR. The TouCAN clears this bit once the internal reset cycle is completed. 0 = Soft reset cycle completed 1 = Soft reset cycle initiated

Table 10-13 TCNMCR Bit Settings (Continued)

Bit(s)	Name	Description
8	FRZACK	TouCAN disable. When the TouCAN enters debug mode, it sets the FRZACK bit. This bit should be polled to determine if the TouCAN has entered debug mode. When debug mode is exited, this bit is negated once the TouCAN prescaler is enabled. This is a read-only bit. 0 = The TouCAN has exited debug mode and the prescaler is enabled. 1 = The TouCAN has entered debug mode, and the prescaler is disabled.
7	SUPV	Supervisor/user data space. The SUPV bit places the TouCAN registers in either supervisor or user data space. 0 = Registers with access controlled by the SUPV bit are accessible in either user or supervisor privilege mode. 1 = Registers with access controlled by the SUPV bit are restricted to supervisor mode.
6	SELFWAKE	Self wake enable. This bit allows the TouCAN to wake up when bus activity is detected after the STOP bit is set. If this bit is set when the TouCAN enters low-power stop mode, the TouCAN will monitor the bus for a recessive to dominant transition. If a recessive to dominant transition is detected, the TouCAN immediately clears the STOP bit and restarts its clocks. If a write to CANMCR with SELFWAKE set occurs at the same time a recessive-to-dominant edge appears on the CAN bus, the bit will not be set, and the module clocks will not stop. The user should verify that this bit has been set by reading CANMCR. Refer to **10.6.2 Low-Power Stop Mode** for more information on entry into and exit from low-power stop mode. 0 = Self wake disabled. 1 = Self wake enabled.
5	APS	Auto power save. The APS bit allows the TouCAN to automatically shut off its clocks to save power when it has no process to execute, and to automatically restart these clocks when it has a task to execute without any CPU intervention. 0 = Auto power save mode disabled; clocks run normally. 1 = Auto power save mode enabled; clocks stop and restart as needed.
4	STOPACK	Stop acknowledge. When the TouCAN is placed in low-power stop mode and shuts down its clocks, it sets the STOPACK bit. This bit should be polled to determine if the TouCAN has entered low-power stop mode. When the TouCAN exits low-power stop mode, the STOPACK bit is cleared once the TouCAN's clocks are running. 0 = The TouCAN is not in low-power stop mode and its clocks are running. 1 = The TouCAN has entered low-power stop mode and its clocks are stopped
3:0	IARB[3:0]	Interrupt Arbitration ID. The IARB field is used to arbitrate between simultaneous interrupt requests of the same priority. Each module that can generate interrupt requests must be assigned a unique, non-zero IARB field value.

10.8.2 TouCAN Interrupt Configuration Register

CANICR — TouCAN Interrupt Configuration Register **0xYF FA84**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED					ILCAN[2:0]			IVBA[2:0]			RESERVED				
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

NOTE

If the TouCAN issues an interrupt request after reset and before IVBA[2:0] is initialized, it will drive 0x0F as the "uninitialized" interrupt vector in response to a CPU32 interrupt acknowledge cycle, regardless of the specific event.

Table 10-14 CANICR Bit Settings

Bit(s)	Name	Description
15:11	—	Reserved
10:8	ILCAN[2:0]	When the TouCAN generates an interrupt request, ILCAN[2:0] determines which of the interrupt request signals is asserted. When a request is acknowledged, the TouCAN compares ILCAN[2:0] to a mask value supplied by the CPU32 to determine whether to respond. ILCAN[2:0] must have a value in the range of 0x0 (interrupts disabled) to 0x7 (highest priority).
7:5	IVBA[2:0]	The interrupt vector base address specifies the high-order three bits of all the vector numbers generated by the different TouCAN interrupt sources.
4:0	—	Reserved

10.8.3 Control Register 0

CANCTRL0 — Control Register 0 **0xYF FA86**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
BOFF MSK	ERR MSK	RESERVED		RXMOD		TXMODE		CANCTRL1							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Table 10-15 CANCTRL0 Bit Settings

Bit(s)	Name	Description
15	BOFFMSK	Bus off interrupt mask. The BOFF MASK bit provides a mask for the bus off interrupt. 0 = Bus off interrupt disabled. 1 = Bus off interrupt enabled.
14	ERRMSK	Error interrupt mask. The ERRMSK bit provides a mask for the error interrupt. 0 = Error interrupt disabled. 1 = Error interrupt enabled.
13:12	—	Reserved
11:10	RXMODE	Receive pin configuration control. These bits control the configuration of the CANRX0 and CANRX1 pins. Refer to the **Table 10-16**.
9:8	TXMODE	Transmit pin configuration control. This bit field controls the configuration of the CANTX0 and CANTX1 pins. Refer to **Table 10-17**.
7:0	CANCTRL1	See **Table 10-18**.

Table 10-16 RX MODE[1:0] Configuration

Pin	RX1	RX0	Receive Pin Configuration
CANRX1	0	X	A logic 0 on the CANRX1 pin is interpreted as a dominant bit; a logic 1 on the CANRX1 pin is interpreted as a recessive bit
	1	X	A logic 1 on the CANRX1 pin is interpreted as a dominant bit; a logic 0 on the CANRX1 pin is interpreted as a recessive bit
CANRX0	X	0	A logic 0 on the CANRX0 pin is interpreted as a dominant bit; a logic 1 on the CANRX0 pin is interpreted as a recessive bit
	X	1	A logic 1 on the CANRX0 pin is interpreted as a dominant bit; a logic 0 on the CANRX0 pin is interpreted as a recessive bit

Table 10-17 Transmit Pin Configuration

TXMODE[1:0]	Transmit Pin Configuration
00	Full CMOS[1]; positive polarity (CANTX0 = 0, CANTX1 = 1 is a dominant level)
01	Full CMOS; negative polarity (CANTX0 = 1, CANTX1 = 0 is a dominant level)
1X	Open drain[2]; positive polarity

NOTES:
1. Full CMOS drive indicates that both dominant and recessive levels are driven by the chip.
2. Open drain drive indicates that only a dominant level is driven by the chip. During a recessive level, the CANTX0 and CANTX1 pins are disabled (three stated), and the electrical level is achieved by external pull-up/pull-down devices. The assertion of both TX mode bits causes the polarity inversion to be cancelled (open drain mode forces the polarity to be positive).

10.8.4 Control Register 1

CANCTRL1 — Control Register 1 **0xYF FA86**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CANCTRL0								SAMP	RE-SERVED	TSYNC	LBUF	0D	PROPSE		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-18 CANCTRL1 Bit Settings

Bit(s)	Name	Description
15:8	CANCTRL0	See Table 10-15
7	SAMP	Sampling mode. The SAMP bit determines whether the TouCAN module will sample each received bit one time or three times to determine its value. 0 = One sample, taken at the end of phase buffer segment 1, is used to determine the value of the received bit. 1 = Three samples are used to determine the value of the received bit. The samples are taken at the normal sample point and at the two preceding periods of the S-clock.
6	—	Reserved
5	TSYNC	Timer synchronize mode. The TSYNC bit enables the mechanism that resets the free-running timer each time a message is received in message buffer 0. This feature provides the means to synchronize multiple TouCAN stations with a special “SYNC” message (global network time). 0 = Timer synchronization disabled. 1 = Timer synchronization enabled. **NOTE**: there can be a bit clock skew of four to five counts between different TouCAN modules that are using this feature on the same network.
4	LBUF	Lowest buffer transmitted first. The LBUF bit defines the transmit-first scheme. 0 = Message buffer with lowest ID is transmitted first. 1 = Lowest numbered buffer is transmitted first.
3	—	Reserved
2:0	PROPSEG	Propagation segment time. PROPSEG defines the length of the propagation segment in the bit time. The valid programmed values are 0 to 7. The propagation segment time is calculated as follows: Propagation Segment Time = (PROPSEG + 1) Time Quanta where 1 Time Quantum = 1 Serial Clock (S-Clock) Period

10.8.5 Prescaler Divide Register

PRESDIV — Prescaler Divide Register **0xYF FA88**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
PRESDIV								CANCTRL2							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Table 10-19 PRESDIV Bit Settings

Bit(s)	Name	Description
15:8	PRESDIV	Prescaler divide factor. PRESDIV determines the ratio between the system clock frequency and the serial clock (S-clock). The S-clock is determined by the following calculation: $\text{S-clock} = \frac{f_{sys}}{\text{PRESDIV} + 1}$ The reset value of PRESDIV is 0x00, which forces the S-clock to default to the same frequency as the system clock. The valid programmed values are 0 through 255.
7:0	CANCTRL2	See Table 10-20.

10.8.6 Control Register 2

CANCTRL2 — Control Register 2 **0xYF FA88**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
PRESDIV								RJW		PSEG			PSEG2		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Table 10-20 CANCTRL2 Bit Settings

Bit(s)	Name	Description
15:8	PRESDIV	See Table 10-19.
7:6	RJW	Resynchronization jump width. The RJW field defines the maximum number of time quanta a bit time may be changed during resynchronization. The valid programmed values are 0 through 3. The resynchronization jump width is calculated as follows: Resynchronizaton Jump Width = (RJW + 1) Time Quanta
5:3	PSEG1	PSEG1[2:0] — Phase buffer segment 1. The PSEG1 field defines the length of phase buffer segment 1 in the bit time. The valid programmed values are 0 through 7. The length of phase buffer segment 1 is calculated as follows: Phase Buffer Segment 1 = (PSEG1 + 1) Time Quanta
2:0	PSEG2	PSEG2 — Phase Buffer Segment 2. The PSEG2 field defines the length of phase buffer segment 2 in the bit time. The valid programmed values are 0 through 7. The length of phase buffer segment 2 is calculated as follows: Phase Buffer Segment 2 = (PSEG2 + 1) Time Quanta

10.8.7 Free Running Timer

TIMER — Free Running Timer Register **0xYF FA8A**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
TIMER															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Table 10-21 TIMER Bit Settings

Bit(s)	Name	Description
15:0	TIMER	The free running timer counter can be read and written by the CPU. The timer starts from zero after reset, counts linearly to 0xFFFF, and wraps around. The timer is clocked by the TouCAN bit-clock. During a message, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it increments at the nominal bit rate. The timer value is captured at the beginning of the identifier field of any frame on the CAN bus. The captured value is written into the “time stamp” entry in a message buffer after a successful reception or transmission of a message.

10.8.8 Receive Global Mask Registers

RXGMSKHI — Receive Global Mask Register High **0xYF FA90**
RXGMSKLO — Receive Global Mask Register Low **0xYF FA92**

MSB 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	LSB 16
MID28	MID27	MID26	MID25	MID24	MID23	MID22	MID21	MID20	MID19	MID18	0	1	MID17	MID16	MID15
RESET:															
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MID14	MID13	MID12	MID11	MID10	MID9	MID8	MID7	MID6	MID5	MID4	MID3	MID2	MID1	MID0	0
RESET:															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Table 10-22 RXGMSKHI, RXGMSKLO Bit Settings

Bit(s)	Name	Description
31:0	MIDx	The receive global mask registers use four bytes. The mask bits are applied to all receive-identifiers, excluding receive-buffers 14 and 15, which have their own specific mask registers. Base ID mask bits MID[28:18] are used to mask standard or extended format frames. Extended ID bits MID[17:0] are used to mask only extended format frames. The RTR/SRR bit of a received frame is never compared to the corresponding bit in the message buffer ID field. However, remote request frames (RTR = 1) once received, are never stored into the message buffers. RTR mask bit locations in the mask registers (bits 20 and 0) are always zero, regardless of any write to these bits. The IDE bit of a received frame is always compared to determine if the message contains a standard or extended identifier. Its location in the mask registers (bit 19) is always one, regardless of any write to this bit.

10.8.9 Receive Buffer 14 Mask Registers

RX14MSKHI — Receive Buffer 14 Mask Register High **0xYF FA94**
RX14MSKLO — Receive Buffer 14 Mask Register Low **0xYF FA96**

The receive buffer 14 mask registers have the same structure as the receive global mask registers and are used to mask buffer 14.

10.8.10 Receive Buffer 15 Mask Registers

RX15MSKHI — Receive Buffer 15 Mask Register High **0xYF FA98**
RX15MSKLO — Receive Buffer 15 Mask Register Low **0xYF FA9A**

The receive buffer 15 mask registers have the same structure as the receive global mask registers and are used to mask buffer 15.

10.8.11 Error and Status Register

ESTAT — Error and Status Register **0xYF FAA0**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
BITER		ACK ERR	CRC ERR	FORM ERR	STUFF ERR	TX WARN	RX WARN	IDLE	TX/RX	FCS		0	BOFF INT	ERR INT	WAKE INT
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register reflects various error conditions, general status, and has the enable bits for three of the TouCAN interrupt sources. The reported error conditions are those which have occurred since the last time the register was read. A read clears these bits to zero.

Table 10-23 ESTAT Bit Settings

Bit(s)	Name	Description
15:14	BITERR	Transmit bit error. The BITERR[1:0] field is used to indicate when a transmit bit error occurs. Refer to Table 10-24. **NOTE**: The transmit bit error field is not modified during the arbitration field or the ACK slot bit time of a message, or by a transmitter that detects dominant bits while sending a passive error frame.
13	ACKERR	Acknowledge error. The ACKERR bit indicates whether an acknowledgment has been correctly received for a transmitted message. 0 = No ACK error was detected since the last read of this register. 1 = An ACK error was detected since the last read of this register.
12	CRCERR	Cyclic redundancy check error. The CRCERR bit indicates whether or not the CRC of the last transmitted or received message was valid. 0 = No CRC error was detected since the last read of this register. 1 = A CRC error was detected since the last read of this register.
11	FORMERR	Message format error. The FORMERR bit indicates whether or not the message format of the last transmitted or received message was correct. 0 = No format error was detected since the last read of this register. 1 = A format error was detected since the last read of this register.
10	STUFERR	Bit stuff error. The STUFFERR bit indicates whether or not the bit stuffing that occurred in the last transmitted or received message was correct. 0 = No bit stuffing error was detected since the last read of this register. 1 = A bit stuffing error was detected since the last read of this register.
9	TXWARN	Transmit error status flag. The TXWARN status flag reflects the status of the TouCAN transmit error counter. 0 = Transmit error counter < 96. 1 = Transmit error counter ≥ 96.

Table 10-23 ESTAT Bit Settings (Continued)

Bit(s)	Name	Description
8	RXWARN	Receiver error status flag. The RXWARN status flag reflects the status of the TouCAN receive error counter. 0 = Receive error counter < 96. 1 = Receive error counter ≥ 96.
7	IDLE	Idle status. The IDLE bit indicates when there is activity on the CAN bus. 0 = The CAN bus is not idle. 1 = The CAN bus is idle.
6	TX/$\overline{RX}$	Transmit/receive status. The TX/RX bit indicates when the TouCAN module is transmitting or receiving a message. TX/RX has no meaning when IDLE = 1. 0 = The TouCAN is receiving a message if IDLE = 0. 1 = The TouCAN is transmitting a message if IDLE = 0.
5:4	FCS	Fault confinement state. The FCS[1:0] field describes the state of the TouCAN. Refer to **Table 10-25**. If the SOFTRST bit in CANMCR is asserted while the TouCAN is in the bus off state, the error and status register is reset, including FCS[1:0]. However, as soon as the TouCAN exits reset, FCS[1:0] bits will again reflect the bus off state. Refer to **10.4.4 Error Counters** for more information on entry into and exit from the various fault confinement states.
3	—	Reserved
2	BOFFINT	Bus off interrupt. The BOFFINT bit is used to request an interrupt when the TouCAN enters the bus off state. 0 = No bus off interrupt requested. 1 = When the TouCAN state changes to bus off, this bit is set, and if the BOFFMSK bit in CANCTRL0 is set, an interrupt request is generated. This interrupt is not requested after reset.
1	ERRINT	Error Interrupt. The ERRINT bit is used to request an interrupt when the TouCAN detects a transmit or receive error. 0 = No error interrupt request. 1 = If an event which causes one of the error bits in the error and status register to be set occurs, the error interrupt bit is set. If the ERRMSK bit in CANCTRL0 is set, an interrupt request is generated. To clear this bit, first read it as a one, then write as a zero. Writing a one has no effect.
0	WAKEINT	Wake interrupt. The WAKEINT bit indicates that bus activity has been detected while the TouCAN module is in low-power stop mode. 0 = No wake interrupt requested. 1 = When the TouCAN is in low-power stop mode and a recessive to dominant transition is detected on the CAN bus, this bit is set. If the WAKEMSK bit is set in CANMCR, an interrupt request is generated.

Table 10-24 Transmit Bit Error Status

BITERR[1:0]	Bit Error Status
00	No transmit bit error
01	At least one bit sent as dominant was received as recessive
10	At least one bit sent as recessive was received as dominant
11	Not used

Table 10-25 Fault Confinement State Encoding

FCS[1:0]	Bus State
00	Error active
01	Error passive
1X	Bus off

10.8.12 Interrupt Mask Register

IMASK — Interrupt Mask Register **0xYF FAA2**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
IMASKH								IMASKL							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-26 IMASK Bit Settings

Bit(s)	Name	Description
15:8, 7:0	IMASKH, IMASKL	IMASK contains two 8-bit fields, IMASKH and IMASKL. IMASK can be accessed with a 16-bit read or write, and IMASKH and IMASKL can be accessed with byte reads or writes. IMASK contains one interrupt mask bit per buffer. It allows the CPU to designate which buffers will generate interrupts after successful transmission/reception. Setting a bit in IMASK enables interrupt requests for the corresponding message buffer.

10.8.13 Interrupt Flag Register

IFLAG — Interrupt Flag Register **0xYF FAA4**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
IFLAGH								IFLAGL							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-27 IFLAG Bit Settings

Bit(s)	Name	Description
15:8, 7:0	IFLAGH, IFLAGL	IFLAG contains two 8-bit fields, IFLAGH and IFLAGL. IFLAG can be accessed with a 16-bit read or write, and IFLAGH and IFLAGL can be accessed with byte reads or writes. IFLAG contains one interrupt flag bit per buffer. Each successful transmission/reception sets the corresponding IFLAG bit and, if the corresponding IMASK bit is set, an interrupt request will be generated. To clear an interrupt flag, first read the flag as a one, and then write it as a zero. Should a new flag setting event occur between the time that the CPU reads the flag as a one and writes the flag as a zero, the flag is not cleared. This register can be written to zeros only.

10.8.14 Error Counters

RXECTR — Receive Error Counter **0xYF FAA6**
TXECTR — Transmit Error Counter **0xYF FAA6**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RXECTR								TXECTR							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-28 RXECTR, TXECTR Bit Settings

Bit(s)	Name	Description
15:8, 7:0	RXECTR, TXECTR	Both counters are read only, except when the TouCAN is in test or debug mode.

SECTION 11
DATA LINK CONTROLLER MODULE (DLCMD2)

11.1 Scope

This document contains data for the data link controller digital module (DLCMD2). This module is based on the IMB DLCMD module, but has several feature enhancements including those required for the IMB3. The primary purpose of this document is to form the foundation for the functionality and features of the module. This module is designed in a modular structure and is fully compatible with the inter-module bus version 3.

The DLCMD2 is designed with sufficient flexibility to accommodate feature mixes such as byte or symbol-level message buffering.

11.2 Features

The DLCMD2 is essentially the digital portion of a Class B serial data link controller. A separate transceiver is required. The DLCMD2 will provide the following features:

- SAE J1850 compatible
- GM class 2 compatible
- 10.4 Kbytes/s VPW bit format
- Handles all network protocol functions (access, arbitration, error detection)
- Parallel 16-bit accesses
- All registers are individually addressable
- Polling and IMB3 interrupt generation with vector lookup available
- Transmit buffer first byte can be loaded without a command byte
- Message buffering on transmit and receive
- 8-bit hardware CRC generation and checking
- No on-board oscillator (uses system clock)
- DLCMD2 logic is clocked from IMB3
- No analog (transceiver) circuit
- Interface to the external transceiver
- Transmit and receive block mode supported
- Transmit and receive 4X mode supported
- Two extra 1-bits sent if lose arbitration on a byte boundary
- Transmitter underrunning indication added to status register
- IMB3 full feature support with option plug selection
- Software programmable prescaler to support two 128-MHz system clock range
- Programmable receiver input polarity
- Programmable normalization bit format
- Digitally filtered receiver
- Power conserving sleep mode with wakeup from bus activity and no loss of data
- IFR type 1, 2 and 3 supported

- Auto retry for loss of arbitration and errors
- Symbol timing control and pre-scaler register
- Symbol timing data register (SDATA)
- Write access to symbol timing parameter table through SDATA

11.3 Background

The DLCMD2 is an evolution of an earlier module found on IMB MCUs.The analog function, or transceiver, necessary to interface to the J1850 bus will be applied external to the module, and will not be built into the DLCMD2.

11.4 Applicable Documents

- SAE J1850 Class B Data Communications Network Interface

11.5 General Requirements

This module is an integrated module for inclusion on-board an IMB3 MCU. Refer to **Table E-20**.

11.6 Logic Description

The data link controller module (DLCMD2) provides access to an external serial communication multiplex bus, operating according to the SAE J1850 protocol.

This section describes the features, functions, and operation, of the DLCMD2 used as one of several nodes in a vehicle multiplex and/or diagnostic wiring network. All control, status, and message bytes (head of FIFOs only) are accessible as memory mapped registers within the MCU.

The DLCMD2 module supports the SAE J1850 protocol. It retains the maximum throughput performance for single-chip applications including full J1850 message-level buffering but does not provide an internal transceiver.

11.6.1 Block Diagram

A block diagram of the DLCMD2 is shown in **Figure 11-1**.

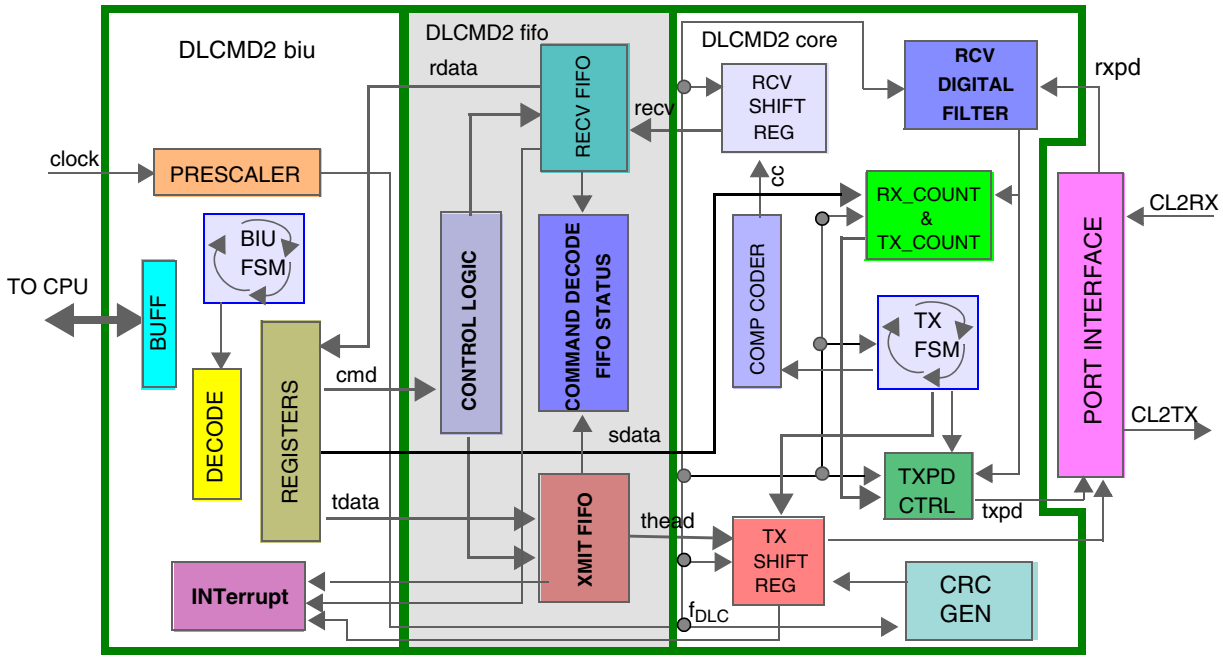


Figure 11-1 DLCMD2 Block Diagram

11.6.2 DLCMD2 Operation

The sections that follow describe the operation of the DLCMD2.

11.6.2.1 General

The DLCMD2 handles J1850 messages with minimal CPU servicing. The MCU will typically transfer complete messages into the DLCMD2 for transmission on the J1850 data link and is interrupted only when a complete message is received from the J1850 data link. Internal buffers of 20 bytes on the receive side and 11 bytes on the transmit side allow full message length operations (maximum 12 bytes normal mode, including a one-byte CRC). The DLCMD2 handles all arbitration and error detection duties internally.

The class 2 data link has been defined as the GM implementation of the SAE J1850 automotive communications protocol. The class 2 bus is a 10.4 Kbytes/s carrier sense multiple access with collision resolution (CSMA/CR) communications bus. CSMA/CR operates by arbitrating ownership of the bus on a bit-by-bit basis. The major advantage of CSMA/CR is that no message is lost in a collision. If a node determines that it needs to send a message while another message is in progress, it must wait until the link is idle. When one node begins to transmit (after bus idle), all nodes desiring to transmit are obligated to begin their transmission at the same time. The rule of arbitration is that any node that transmits a 1-bit when another node transmits a 0-bit stops transmitting on the bus immediately. This is called a zero-dominant bus. See **11.7 Signals Over-**

view for 1-bit and 0-bit definitions. All nodes are obligated to receive all bits of every message on the bus, even while transmitting. Arbitration continues to the end of a message, if necessary, resulting in a properly transmitted message even if arbitration were to continue to the end of the message. When the bus becomes idle, the node(s) which previously lost arbitration will re-transmit its message if the TxFIFO was not cleared or overwritten with another message.

11.6.2.2 Logic Section Description and Relation to Transceiver

The logic section includes the IMB3 interface, J1850 waveform generation and timing logic, buffers for data (transmit and receive), error detection code generation and checking, configuration logic, control logic, status logic, and arbitration logic.

The nondestructive contention protocol of the J1850 bus requires that there be an active and a passive state of the bus. The bus is in the active or driven state when one or more of the connected transceivers is active and passive when all transceivers are inactive. This is a logical wired OR arrangement.

The function of the transceiver is to drive the bus active in response to a signal from the DLCMD2 logic and to detect the state of the bus for the DLCMD2 logic handler. The transceiver establishes and reliably detects the state of the bus within a limited period of time. It does so in the presence of conducted and induced noise and without creating radio interference. Operation of the transceiver is constrained by available power and the need to tolerate a number of abnormal conditions.

The J1850 bus is intended to work in a relatively noisy environment. The main source of low frequency noise is ground offset between the nodes. Proper operation is assured with any combination of ground offsets up to a maximum differential of two volts at any frequency. Induced noise tends to be short duration pulses. The protocol handler includes a digital filter to remove these pulses. Additional filtering is not needed in the receiver which responds quickly and avoids stretching large amplitude pulses.

11.6.2.3 DLCMD2 Transmit/Receive Operation

A standard data exchange is composed of one data byte and (in some cases) one command byte going from CPU to DLCMD2 or one status byte and one data byte going from DLCMD2 to CPU. The use of the byte written to the DLCMD2 is specified in the command byte that accompanies it or deduced from current DLCMD2 state. The command byte also contains instructions for the DLCMD2 regarding the receive first in/first out (FIFO) buffer, transmit actions, resetting the transmitter, and sending a break signal. The status byte that the DLCMD2 sends to the CPU contains information on the status of the receive FIFO buffer, the status of the transmit FIFO buffer, the condition of the bus, and the type of accompanying data. The data accompanying the status byte can be data received off of the J1850 bus, a completion code which contains information about a received message, or nothing.

Figure 11-2 shows the DLCMD2 transmit/receive operation.

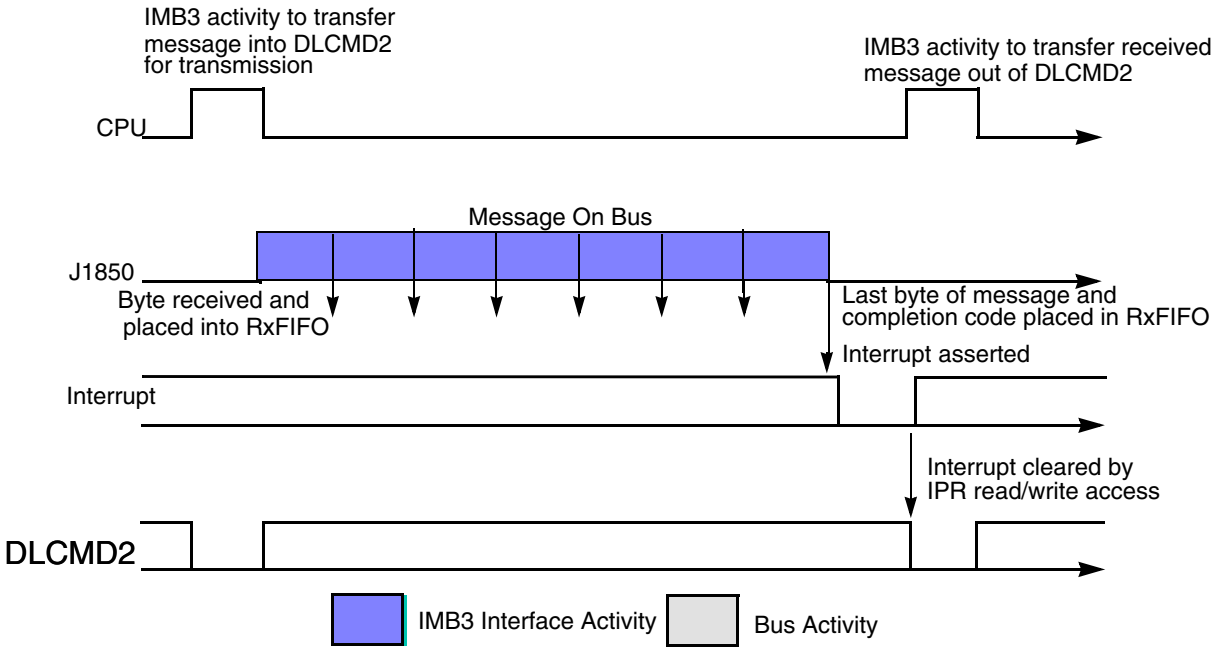


Figure 11-2 Typical Transmit/Receive Operation

All messages received from the J1850 bus will have their start bit removed and the CRC replaced with a completion code. All other bytes of the message are placed in order, most significant bit (MSB) first, in the receive FIFO buffer.

The DLCMD2 requests servicing by requesting an interrupt. Interrupts may be selected to operate in two different modes or be disabled. Typically, the DLCMD2 will only interrupt the CPU when a complete message has been received. The CPU then will service the DLCMD2 and remove the message. When the CPU desires to send a message on the J1850 bus, it will select the DLCMD2 and transfer a complete message (without start bit or CRC).

The DLCMD2 allows the CPU to continually write message bytes to be transmitted without intervening command bytes; only the last byte must be accompanied by a command byte indicating “last byte.” User code must read both status and receive data in pairs. The DLCMD2 supports aligned word writes and reads of certain locations.

The DLCMD2 can be programmed by the CPU to enter a power conserving sleep mode as soon as bus traffic stops. If interrupts are enabled, the DLCMD2 will wake-up its internal circuitry and interrupt the host when activity on the bus is sensed. The DLCMD2 will be able to correctly receive the first message that wakes it up in sleep mode.

Error conditions and transmit status, such as lost arbitration, are sent to the CPU either in the status byte or in the completion code that is placed in the receive FIFO immediately after each received message.

11.6.2.4 Message Transmission

As described in the preceding section, the DLCMD2 is loaded with a message from the CPU for transmission. The DLCMD2 will then add a start bit to the outgoing data, and contend for a message slot on the J1850 bus. The transmit buffer in the DLCMD2 is 11 bytes long, to allow complete messages to be transferred to the DLCMD2 for transmission. Information about which byte the CPU is transferring is sent to the DLCMD2 as part of the control information sent with each byte transferred. When the DLCMD2 has transmitted all of the message bytes, it will automatically append a CRC to the end of the message.

The DLCMD2 will automatically retry a transmission if it lost arbitration. The auto retry feature causes the DLCMD2 to signal to the CPU (via the status byte) that the transmit FIFO is full until the message is successfully sent. As soon as the CPU transfers the last byte of a message, the DLCMD2 will indicate that the transmit FIFO buffer (TxFIFO) is full. Once successfully sent, the DLCMD2 will signal the CPU that the TxFIFO is empty and hence ready for the next message. Each time the message loses arbitration, the completion code for that received message will indicate that the transmitter attempted transmission, and lost arbitration. As soon as a transmit slot on the bus becomes open, the DLCMD2 will automatically attempt to retransmit the message. If there were any errors during the transmission of the message the auto retry feature will cause the message to be retransmitted. The auto retry feature can be terminated by the CPU through the command byte. This causes the DLCMD2 to finish its current transmit activity, and then clear the transmit buffer. If there is no transmit activity when the auto retry is disabled and the DLCMD2 previously attempted to transmit, the DLCMD2 will immediately clear the transmit buffer.

11.6.2.5 Message Reception

Receiving information off of the J1850 bus occurs in much the same manner as sending data.

NOTE

By definition, every message a DLCMD2 sends on the data link is also received by its own receiver as if the transmission had been initiated by a different node.

The DLCMD2 will automatically remove the start bit from the message and check the CRC for errors. If a CRC error occurs, it is flagged in the message completion information (completion code) that takes the place of the CRC byte in the receive FIFO buffer (RxFIFO). The DLCMD2 will interrupt the CPU to signal that a complete message has been received or when the RxFIFO is approaching full. When the CPU starts the transfer process, the status byte from the DLCMD2 indicates the data's presence and the amount of data left in the RxFIFO. When a DLCMD2 loses arbitration to another node, it will continue to receive the remainder of the message. The completion

code will indicate that the DLCMD2 contended for a transmit slot, and lost arbitration to the received message. The CPU does not need to resend the message if the auto retry feature is enabled.

The timing of the transmit waveform is re-synchronized on each edge as received off of the bus.

Provisions have been made for immediate in-message reply to allow a path for compatibility with other J1850 implementations. In-frame response (IFR) requires a byte-by-byte interrupt mode and careful CPU attention to accomplish. IFR is described in detail in a later section.

A break/reset waveform on the bus is shaped such that it will win arbitration against any currently transmitting message. When the DLCMD2 senses that such a waveform has occurred on the bus, it will stop transmitting its current message, reset its transmitter (clear the data link controller module (DLCMD2) TxFIFO), and set a bit in the completion code and request an interrupt to indicate to the CPU that such a condition has occurred. Since the break signal always wins arbitration, any in-progress messages will simply lose arbitration, and the DLCMD2 will treat the in-progress received message as complete. If a break occurs while there are no in progress messages, a completion code indicating that a break has occurred will be placed in the RxFIFO and a CPU interrupt generated. The break/reset waveform is sent by a DLCMD2 combination in the command byte. This waveform is sent immediately upon the command byte being latched in. If there is a current transmission, it will be stopped, and the break waveform sent. A break will also take the DLCMD2 out of 4X mode.

11.6.2.6 Sleep Mode

The CPU may put the DLCMD2 in sleep mode by setting the STOP bit in the MCR. Setting this bit will tell the DLCMD2 to halt its internal clocks, immediately after any currently in progress messages are completed.

Interrupts to the host on bus activity can be disabled by configuring the DLCMD2 with the ILR register. Normal use of the sleep feature will have interrupts to the host enabled, so that the host will not miss any messages on the data link. If interrupts are disabled, and then the DLCMD2 is put to sleep, the only way to wake up the DLCMD2 is by the CPU clearing the STOP bit.

11.6.2.7 Debug Mode

This mode is entered from the reset state or from the run state by asserting or deasserting the appropriate signals. See **11.8.5 DLCMD2 DEBUG** for details.

11.6.2.8 4X Speed Mode

The DLCMD2 has the ability to transfer large amounts of data in a 4X speed mode under special conditions such as memory loading at the vehicle factory, and diagnostic responses. There is a bit in the MCR to control this feature. The 4X speed mode affects only the bit timing section of the DLCMD2, including the digital filter. A break will reset

any listening nodes out of 4X speed mode. The 4X speed mode is not for use during normal operation.

To use 4X mode there must be coordination of all nodes on the network. This mode will not work properly at the network level unless ALL nodes are transmitting in 4X mode. Certain nodes may elect not to take part in 4X communications; these nodes may listen but must not transmit.

Notification of entrance into the special 4X mode is communicated to all nodes with a regular speed message indicating the bus protocol speed switch to 4X mode. A BREAK received will automatically take the DLCMD2 out of 4X mode.

11.6.2.9 Block Mode

The DLCMD2 has the ability to receive a message of unlimited length, provided the CPU reads bytes out of the RxFIFO before it overflows.

If the TxFIFO was filled with no last byte indicated, the DLCMD2 will start sending that message in terminate auto-retry mode; the status will indicate “TxFIFO almost full” as soon as the first byte is sent, and “TxFIFO contains some data bytes” as soon as the second byte is sent. As new data is written to the TxFIFO, the status will accurately reflect the condition of the TxFIFO until a “last byte” is written. When a “last byte” command has been sent to the DLCMD2, the TxFIFO will indicate “TxFIFO full” until the transmission is finished. The DLCMD2 will send an infinite length message if properly handled by the CPU.

11.6.2.10 Error Detection

The DLCMD2 uses a digital filter and the cyclical redundancy check (CRC) byte to detect errors.

The digital filter eliminates short duration noise spikes and transition noise from the incoming waveform. It is a “hysteresis” type filter with a time constant of approximately 8 µs, depending on the IMB3 clock frequency. The step response of the filter is a step function delayed by ±8 µs. It can be described by example with a 2-MHz clock (T_{DLC} = 0.5 µs) and a 4 bit up/down counter. The counter counts up for every oscillator clock pulse when the input is in the active state and down when the input is in the passive state. The counter clamps at 0 and 15. The output is defined by **Table 11-1**.

Table 11-1 Digital Filter Output

Count	Output
0	0
1 – 14	Unchanged
15	1

This filter will cause a receive delay of 16-17 times T_{DLC} in addition to the delay in the transmitter and receiver analog interface circuitry. This delay’s only variation is due to the tolerance on the CPU’s oscillator.

In simple terms, the effect of the filter is that a low or high level on the bus is not recognized unless it is longer than about 8 µs.

The CRC byte is used by the receiver to determine if any errors have occurred during transmission. CRC generation uses the divisor polynomial: $X^8 + X^4 + X^3 + X^2 + 1$. The transmitted CRC is generated by the receiver by initially setting the remainder polynomial to all ones, serially processing the first byte and then all remaining bytes of the message, and appending the one's complement of the remainder to the end of the transmitted message. The receiver uses the same divisor polynomial to process all received message bits including the CRC but excluding the start bit. If the transmission is received correctly, at the completion of the message reception, the remainder polynomial will be: $X^7 + X^6 + X^2$ (%11000100 or $C4).

This CRC code will detect all single and 2 bit errors and all 8 bit burst errors (i.e., any number of errors within a single 8-bit span). Severe noise will normally be detected separately as a bit timing error.

11.6.2.11 Arbitration

The J1850 bus is classified as a carrier sense multiple access with collision resolution (CSMA/CR). This type of bus operates by arbitrating ownership of the network on a bit-by-bit basis. The major advantage of CSMA/CR is that no message is lost in a collision. If a node determines that it needs to send a message while another message is in progress, it must wait until the link is idle. When one node begins to transmit (after bus idle), all nodes desiring to transmit are obligated to begin their transmission at the same time. The rule of arbitration is that any node that transmits a 1-bit when another node transmits a 0-bit stops transmitting on the bus immediately. This is called a 0 dominant bus. To prevent noise from corrupting the bus, arbitration is also lost if a 1-bit is detected when a 0-bit was transmitted. All nodes are obligated to receive all bits of every message on the bus, even while transmitting. Arbitration continues to the end of a message, if necessary. If an opposite bit is detected, transmission is immediately stopped unless it occurs on the 8th bit of a byte. In this case the DLCMD2 will automatically append two extra 1-bits and then stop transmitting.

NOTE

Two extra bits must be transmitted due to the fact that the eighth bit of a byte is an active, high level on the J1850 bus. Therefore the first extra bit will be a passive, low level, and one more bit is needed in the active, high level so that after the falling edge of this bit the bus will be in the passive state.

These two extra bits will be arbitrated normally and thus will not interfere with another message.

11.6.2.12 Timebase Generation

The generation of time intervals within the DLCMD2 module takes into account the variations of MCU family, oscillator frequency, and physical interface delays that may occur. The frequency f_{IMB3} is sent to the DLCMD2 where it is further divided by "n" (set

by the user through the **Symbol Timing Control and Pre-Scaler Register (SCTL)**), such that the main DLCMD2 operating frequency (f_{DLC}) is approximately 2.00 MHz, depending on IMB3 clock frequency.)

The DLCMD2 J1850 bit timings are derived from the f_{DLC} time base and a stored table of VPW symbol values. The table of VPW symbol values is generated by the user via the **Symbol Timing Data Register (SDATA**).

NOTE

The f_{DLC} signal defines the fundamental resolution of the DLCMD2 module. All bit timings within the DLCMD2 are based upon integer multiples of the fundamental resolution.

Should a physical interface exhibit an unusually large delay, the length of the J1850 transmit symbol values stored in the DLCMD2 symbol table may be reduced pro rata to compensate.

The VPW symbol length table is determined by the user after the symbol lengths have been set via the transceiver REXT bias resistor selection. The REXT resistor values are chosen so as to minimize the radio frequency interference (RFI) from the J1850 bus by inputting a 10.4-kHz square wave into the transmitter and subsequently out on the J1850 bus. These biasing resistors will affect the length of the VPW symbols to some degree due to their effect on the corners of the bus signal that is output by the transmitter.

11.6.2.13 Receive and Transmit Message Buffers

The RxFIFO and TxFIFO are 20 and 11 bytes in length respectively, to allow buffering of a complete message.

The TxFIFO must be able to differentiate between three types of data:

1. Message data byte
2. First byte in message
3. Last byte in message

The auto retry feature recirculates the bytes of a message in the TxFIFO until the message is successfully sent, at which time the FIFO's contents are flushed. When auto retry is disabled, the FIFO will complete an "in progress" transmission, if any, and then flush the contents of the FIFO. If the node is not transmitting, the FIFO will be flushed immediately. If the auto retry feature is disabled as a message is being loaded into theDLCMD2, the DLCMD2 will try to transmit the message once and then clear the transmit FIFO.

Received bytes will be placed into the RxFIFO as soon as they are completely received off of the bus. When an EOD has occurred on the bus, a completion code will be inserted into the RxFIFO after the last received byte of the message. The CRC byte will be checked by the logic and discarded.

11.6.2.14 Bus Waveforms Generation

The DLCMD2 supports Huntzicker encoding. Each symbol generated by the DLCMD2 will be synchronized with the latest edge seen on the bus. Errors due to oscillator tolerance and ground offsets will not accumulate through the message in this manner. Synchronizing in this manner does require that the bit timing unit account for all known delays. The transmit timing will have a very narrow window due to oscillator tolerance and variation in the known delay only. The receive timing will have much wider windows due to the uncertainty in determining edge position resulting from ground offsets, oscillator tolerance, and delay time variation. In either case, transmit or receive, the timing will be specified as beginning when the DLCMD2 senses a transition, to when the DLCMD2 causes or senses the next transition.

11.6.2.15 Huntzicker Encoding

The information contained in this section describes the bit timing section of the logic on the DLCMD2. The timing of VPW (Huntzicker) waveforms requires knowledge of the fixed delays in the transceiver and the logic section. The J1850 bus is a single wire ground referenced bus. This configuration has two important consequences for the bit timing section. In order to reduce the radiated emissions of the bus, each edge must be slew rate limited, and have its corners shaped. To not adversely affect the corner shaping, the specification must not place limits that force the corners. The other consequence is due to the ground offset requirement for the bus. This requirement dictates a minimal voltage swing necessary to operate in the presence of ground offset. The combination of the two factors gives rise to an uncertainty in both when the receiver (of a receiving node) detects a given transition, and when the transmitter (through its own receiver) detects the same transition.

VPW encoding defines one edge for each symbol. A symbol is composed of a period of time (at a particular state of the bus) and the edge that follows that period. The point of reference for the time period is the trip point that the receiver uses to recognize the preceding transition on the bus. Three independent variables are used to describe the waveform generated. These are the times from the trip point to each of the following transitions threshold levels, and the time between these threshold levels. The corners of the waveform fall outside of the “slew rate” time requirement, and may bargain for time and voltage more freely.

The following symbol limits are consistent with $T_{t,max}$ = 16 µs and an oscillator tolerance of 2%. T_{nom} is the nominal symbol time with no oscillator error and the receiver detecting the transition at $T_{t,max/2}$. T_{r1} to T_{r2} is the required acceptance range while T_{r1typ} to T_{r2typ} is a typical acceptance range with a 2% guard band plus a small margin. T_{r1typ} to T_{r2typ} in **Table 11-2** represent the receiver windows. T_{x1} and T_{x2} in **Table 11-3** represent transmitter windows consistent with a 2% oscillator tolerance and 3 µs for all other variations in the transmit path.

Table 11-2 Receive Windows

Symbol [1, 2, 3]	T_{rnom}	T_{rmin}	T_{rmax}	T_{r1}	T_{r2}	T_{r1typ}	T_{r2typ}	Units
Short 1/0	64	53	75	37	91	34	96	µs
Long 1/0	128	116	141	100	157	96	163	µs
SOF/EOD	200	186	214	170	230	163	239	µs
EOF	280	265	—	249	—	239	320[4]	µs

1. All waveforms less than 8þs will be filtered out by the digital filter, and will not be seen as an error.
2. Break is an active symbol that will be transmitted as at least 239þs in length.
3. All window times include digital as well as analog signal delays.
4. All transmitters are armed as soon as they detect the EOF. The end of the guard band on the EOF serves as a arming point for all transmitters. This is the point that all nodes must have recognized an EOF.

Table 11-3 Transmit Windows

Symbol [1, 2, 3]	T_{xnom}	T_{xmin}	T_{xmax}	Units
Short 1/0	64	60	68	µs
Long 1/0	128	122	134	µs
SOF/EOD	200	193	207	µs
EOF	280	271	289	µs

The symbol waveforms seen on the bus have two important characteristics:

Each transition of the transmitted bus signal, as initiated by CL2TX (LOTI), is slew rate limited and has its corners rounded (wave shaped) so that the nominal rise or fall time is about 16 µs to reduce the radiated emissions of the bus. This wave shaping is disabled when 4X mode is enabled.

The received bus signal needs only a minimal voltage swing around the receiver's nominal trip point voltage for proper detection. The point of reference for the time period is the trip point voltage (V_t) a receiver uses to recognize a transition on the bus and produce the CL2RX (LITO) signal.

The CL2RX signal is digitally filtered with an approximate 8 µs delay at the 10.4 kHz bus rate (2 µs in 4X mode). Since a high or low level input to the filter must last longer than the filter delay time in order to appear at the filter output, noise pulses shorter than this are eliminated.

When a single node is transmitting, the symbol time period between successive transitions is controlled completely by the transmitter's transmit symbol timing logic. When two or more nodes are contending for the bus the start point for an active to passive state transition is determined by the node with the slowest clock rate and the start point for an inactive to active state transition is determined by the node with the fastest clock rate, assuming that both nodes are transmitting the same symbol. The symbol width as controlled by the transmitting node's CL2TX signal, can range from T_{xmin} to T_{xmax}. The receiver's acceptance time window range (T_{rmin} to T_{rmax}) is much broader to allow all widths to be classified into defined symbols.

The time windows are not affected by multiple nodes trying to transmit at the same time during arbitration. This is because one node effectively dominates each transition (the first node to leave the passive state or the last node to leave the active state). Although the fastest or slowest node dominates a particular transition, the arbitration scheme assures that the highest priority message always wins.

J1850 bus transmitter output and input signal waveforms are shown in **Figure 11-3**.

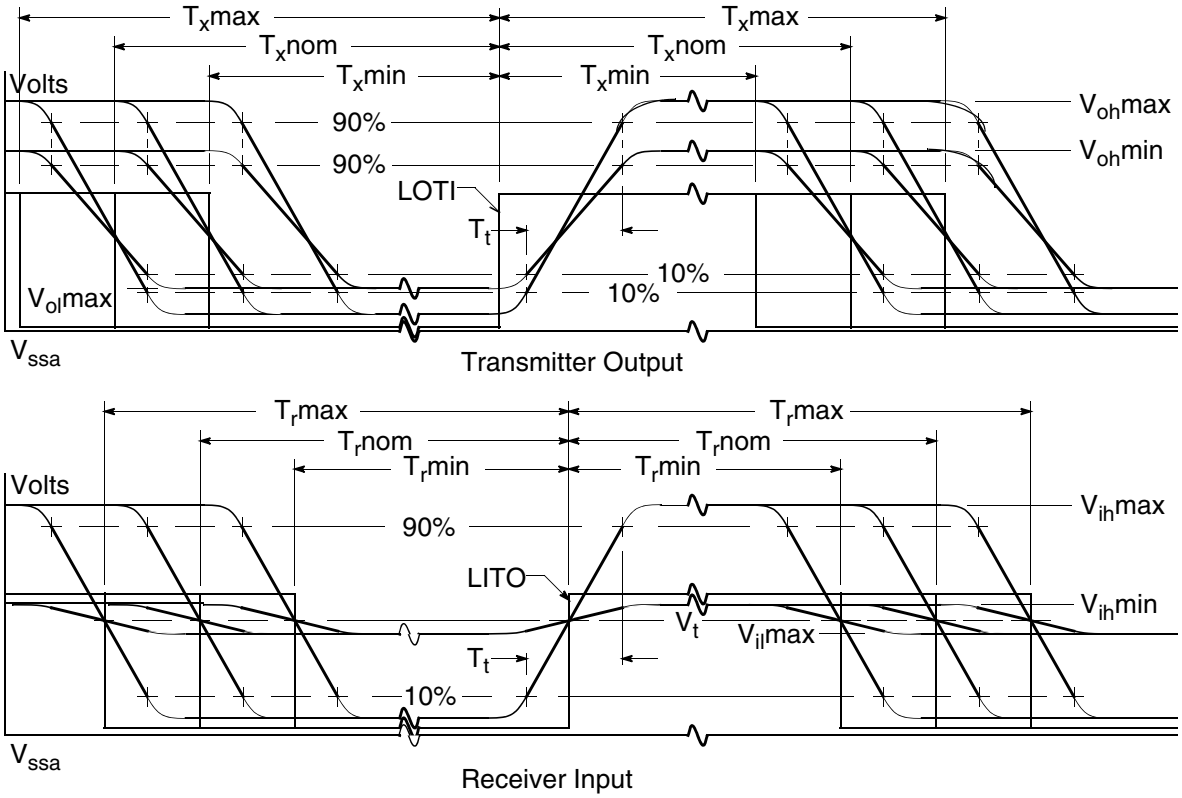


Figure 11-3 VPW Signal Waveforms

NOTES:
1. T_{xmax} is maximum symbol transmission time.
2. T_{xnom} is nominal symbol transmission time.
3. T_{xmin} is minimum symbol transmission time.
4. T_{rmax} is maximum symbol receive window time.
5. T_{rnom} is nominal symbol receive window time.
6. T_{rmin} is minimum symbol receive window time.

11.7 Signals Overview

This section provides an overview of DLCMD2 signals.

11.7.1 J1850 Bus Waveforms

The DLCMD2 module must be able to generate and recognize the set of Huntzicker waveforms described in the following sections. See Figure 11-4. Additionally:

- Each symbol is represented by the time between two consecutive transitions
- There is one transition per symbol and one symbol per transition
- There are both active and passive symbols that are used alternately
- A longer active symbol will dominate a shorter one
- A shorter passive symbol will dominate a longer one

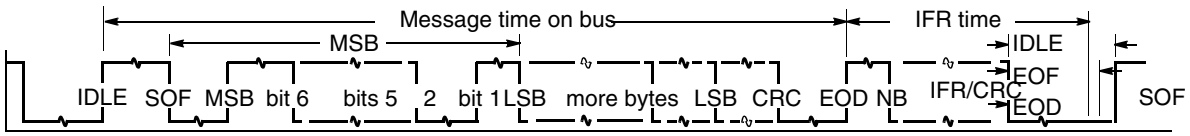


Figure 11-4 Huntzicker Waveform Message

The following sections show the nominal time duration, in microseconds (µs), of the VPW message symbols generated by the DLCMD2 as they appear on the J1850 bus when operating at the normal bus speed. When the DLCMD2 is operating at the high bus speed all 4X symbol times are one fourth that shown, except for “Break”, which will be transmitted the same length in 1X or 4X mode.

11.7.1.1 Start of Frame (SOF)

This active symbol appears at the start of every message when a transmitter drives the bus high to start a message.

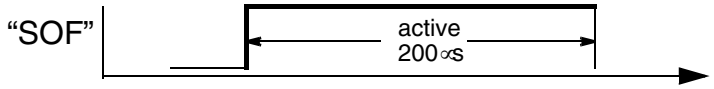


Figure 11-5 Start of Frame Symbol

11.7.1.2 Data Bits

Each data bit is represented by the time between two consecutive transitions. There are both passive and active bit states that are used alternately. The “0” bit is the dominant bit in arbitration.

11.7.1.3 “0” bit

The two dominant “0” bit waveforms are:

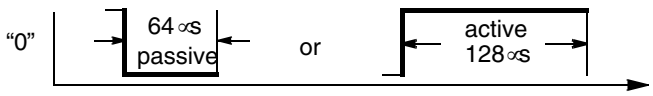


Figure 11-6 Passive “0” and Active “0”

11.7.1.4 “1” Bit

The two “1” bit waveforms are:

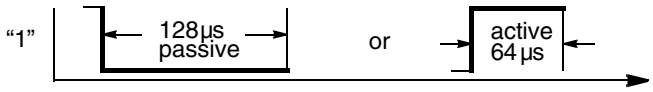


Figure 11-7 Passive “1” and Active “1”

11.7.1.5 End of Data (EOD)

This passive symbol appears after the first CRC byte only in the “request in-frame data” message. It ends when the responding transmitter sends its normalization/format bit prior to the start of the first in-frame response byte. If no node responds, this passive symbol will stretch into an “end of frame” symbol.

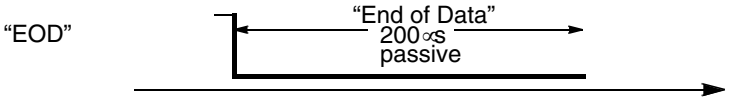


Figure 11-8 End of Data Symbol

11.7.1.6 Normalization Bit

The normalization bit symbol’s duration is the same as an active “1” or “0” bit time. The format of this bit, whether it is a “1” or “0”,” can be selected by the normalization bit format select (NBFS) bit in the **Symbol Timing Control and Pre-Scaler Register (SCTL**). J1850 protocol encourages the use of a “0” when the in-frame response (IFR) ends with a CRC byte and a “1” when the IFR does not end with a CRC byte.

11.7.1.7 End of Frame (EOF)

This passive symbol appears at the end of every message. It is at least 280 µs long. If the bus remains passive until 320 µs, the bus is idle and a transmitter may begin transmitting. If a transmitter desiring bus access detects a rising edge on the bus

between 280 µs and 320 µs (due to clock mismatch between nodes) it may join in and arbitrate.

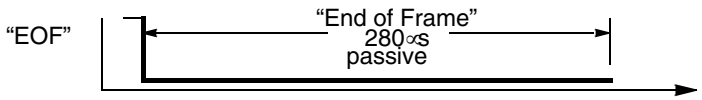


Figure 11-9 End of Frame Symbol

11.7.1.8 Break

The active "Break" signal causes any other transmitting module to stop transmitting immediately because it loses arbitration. It is at least 239 µs long.

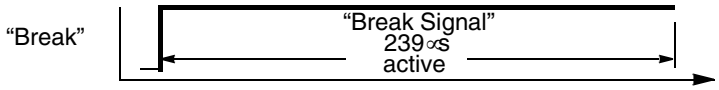


Figure 11-10 Break Symbol Controller Module (DLCMD2)

11.7.2 General Symbol Transmission

The J1850 transmitter will drive the bus to active state and expect that the external RC network will pull the bus back down to the passive state, which is relative since there may be a difference of base ground potential between J1850 nodes in the vehicle. The transmitter is responding to feedback from the receiver in order to know precisely when to switch the transmitter on or off. There is a set of basic transmit timing windows for transmitted symbols within the logic section of the DLCMD2 but if the receiver detects the state of the bus as changing early, the transmitter will also change to that level unless it had not intended to transmit that symbol whereby arbitration is lost and transmission will cease immediately. Thus, all J1850 devices on the J1850 bus synchronize to each other's clock and ground mismatches. Remember that there is only one train of symbols appearing on the bus. The individual symbols are pulled high and released low by various transmitters but the end result is one waveform. It just may be seen differently by the devices due to clock, ground, and power supply variations.

11.7.3 General Symbol Reception

An external transceiver passes unfiltered bus status information to the DLCMD2's Rx pin. Internal to the DLCMD2, the digital 1 or 0 is clocked through a digital delay filter for 16 ticks of its internal frequency clock (a delay of 8 µs at normal (2-MHz f_{DLC}) speed before the filter output changes state. High and low levels on the J1850 bus are timed in the logic section and compared to a set of received symbol threshold windows. Every received high or low level is translated into one of the symbols in the above sections or is flagged as a bit timing error.

The bus is “idle” when the output of the DLCMD2’s digital filter has been in the passive state for 320 µs.

11.7.4 Support For External Transceiver

As shown in Figure 11-11, the DLCMD2 will be designed to use an external (IC) transceiver. Along with the CL2TX and CL2RX signals, a 4XEN signal will be provided at the periphery of the DLCMD2 to disable transceiver waveshaping during DLCMD2 4X mode.

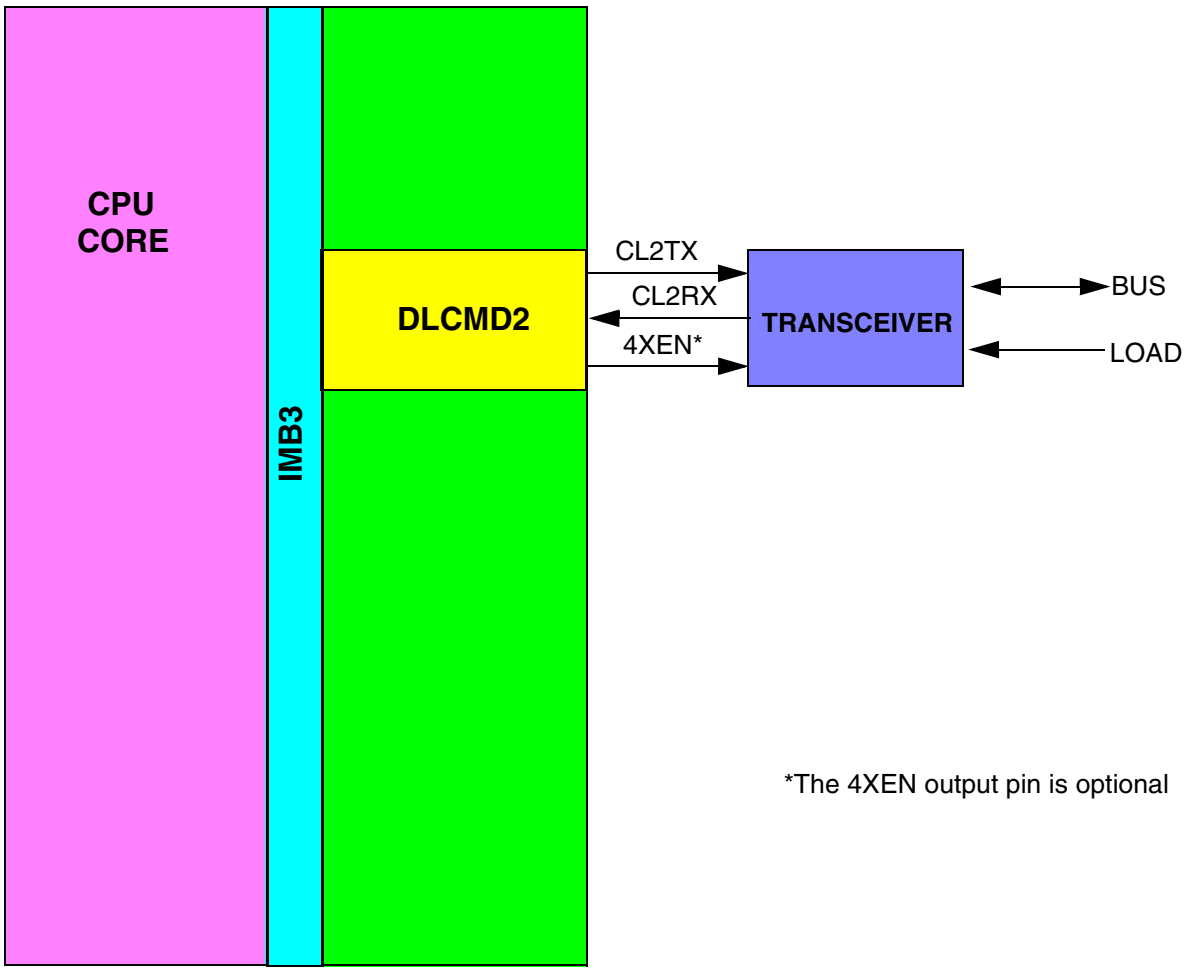


Figure 11-11 Support For External Transceiver

11.8 Operating Modes

This section describes DLCMD2 operating modes. The DLCMD2 has five main modes of operation which interact with the power supplies, pins, and the rest of the MCU. Refer to Figure 11-12.

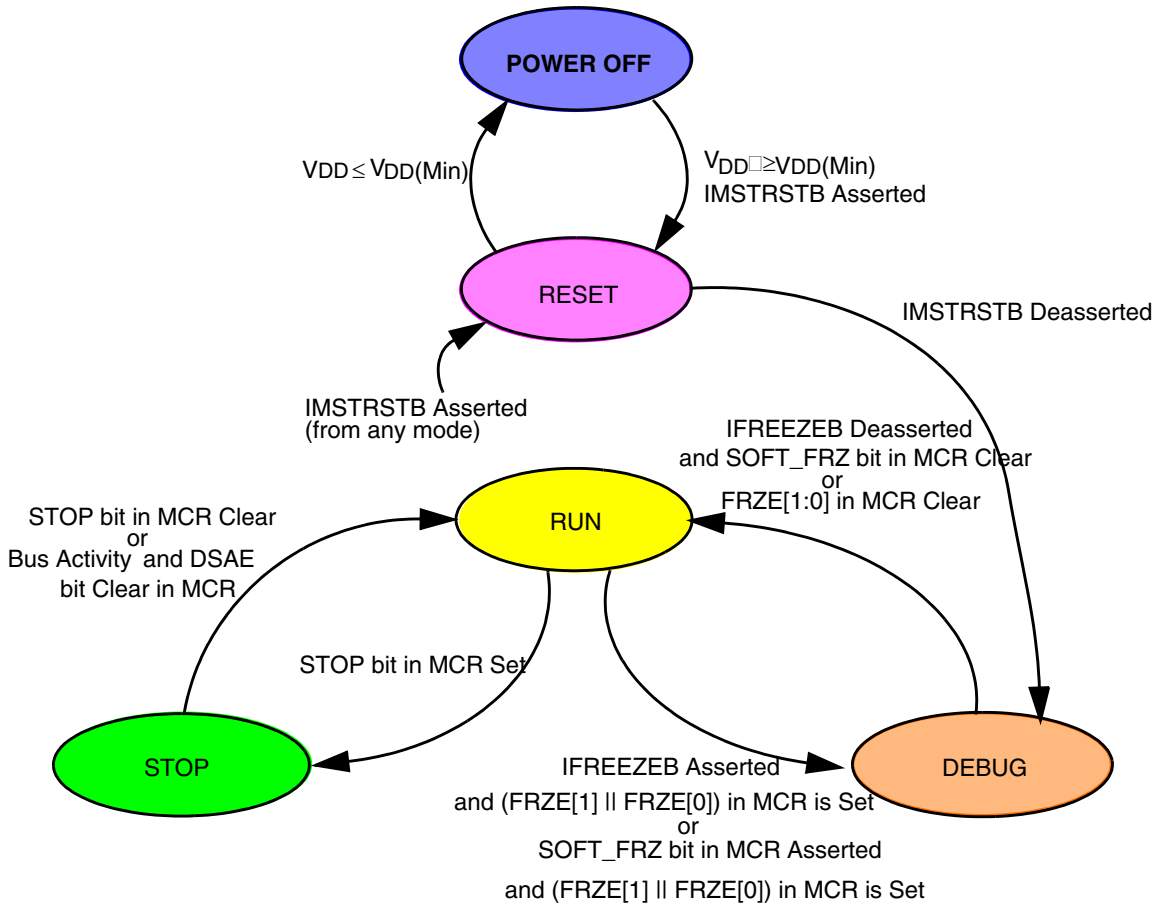


Figure 11-12 DLCMD2 Operation Modes

11.8.1 Power Off

This mode is entered from the reset mode whenever the DLCMD2 supply voltage V_{DD} drops below its minimum specified value for the DLCMD2 to guarantee operation. This implies that the DLCMD2 must be placed in the reset mode before being powered down. In this mode, the pin input and output specifications are not guaranteed.

11.8.2 Reset

This mode is entered from the power off mode whenever the DLCMD2 supply voltage V_{DD} rises above its minimum specified value and IMSTRSTB is asserted. This implies that IMSTRSTB must be asserted while powering up the DLCMD2 or an unknown

state will be entered and correct operation cannot be guaranteed. It is also entered from any other mode on the falling edge of CLOCK after IMSRSTB is asserted.

In this mode V_{DD} is supplied to the internal circuits, which are held in their reset state and the internal DLCMD2 system clock is running. Registers will assume their reset condition. Outputs are held in their programmed reset state, inputs and network activity are ignored.

11.8.3 Run

This mode is entered from the debug mode after all MCU reset sources are no longer asserted. It is entered from the DLCMD2 STOP mode whenever a message is successfully received or the CPU has accessed the DLCMD2 and negated the STOP bit (if previously set).

It is entered from the DLCMD2 LPSTOP mode whenever network activity is sensed although messages will not be received properly until the clocks have stabilized and the CPU is also in the run mode.

In this mode, normal network operation takes place. The user should ensure that all DLCMD2 transmissions have ceased before exiting this mode.

11.8.4 DLCMD2 STOP and LPSTOP

11.8.4.1 DLCMD2 STOP mode

This mode is automatically entered from the run mode whenever the CPU executes a STOP instruction. The IMB3 clocks continue to run, but the CPU clock is stopped.

In this mode, the DLCMD2 internal clocks continue to run and the module will await a valid network message. If a valid network message is successfully received, a CPU interrupt request will be generated (if interrupts are enabled).

Controller module (DLCMD2) DLCMD2 power is only conserved in this mode if the STOP bit in the MCR is set, stopping the DLCMD2 clocks.

11.8.4.2 DLCMD2 LPSTOP mode

This power conserving mode is automatically entered from the run mode whenever the CPU executes a LPSTOP instruction.

In this mode, the DLCMD2 internal clocks are stopped and the module will await any J1850 activity (including noise). If network activity is sensed, then a CPU interrupt request will be generated, restarting both the IMB and DLCMD2 internal clocks.

11.8.5 DLCMD2 DEBUG

This is a special debug mode entered by asserting the SOFT_FRZ bit in the MCR register, or by asserting IMB3 IFREEZEB line. For both, activating the DLCMD2 debug mode is qualified by the FRZE[1:0] in MCR register.

Upon exiting the reset state the SOFT_FRZ bit and FRZE[1:0] bits are set in the MCR register. Hence, reset mode is always followed by the debug mode.

Once this mode is set, the following occurs:

- The pre-scaler divider is stopped, thus halting all related activities.
- Any activity on the J1850 bus will be ignored. The DLCMD2 ignores the CL2RX input pin and drives CL2TX to the passive state.
- The CPU can read and write into most of DLCMD2 registers except for otherwise noted in the register description section.
- The NOT_RDY and FREEZ_ACK bits in MCR register are set.
- After asserting the debug mode configuration bits, the user must wait for the FREEZ_ACK bit to be set in MCR register, before executing any other action to the DLCMD2; otherwise the DLCMD2 may operate in an unpredictable way.

Exiting the debug mode is done in one of the following ways:

- Both, IMB3 freeze and SOFT_FRZ bits are negated.
- CPU negates the FRZE bit.
- Once debug mode is exited, the DLCMD2 is ready to transmit/receive normally on the J1850 bus.

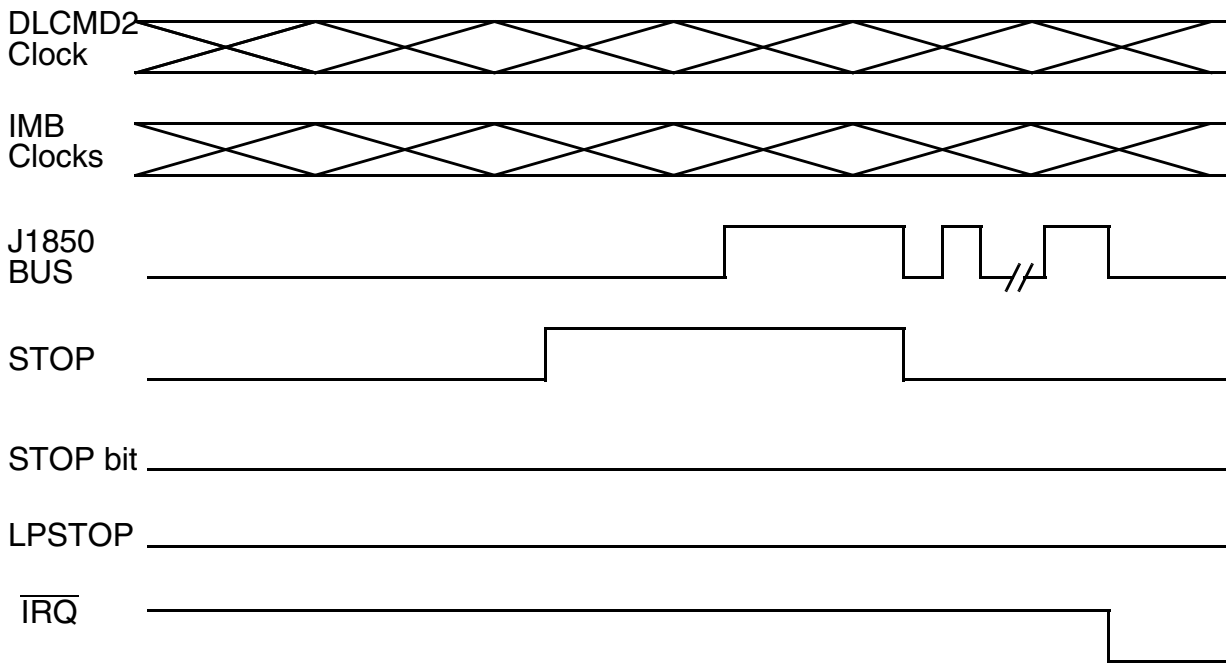


Figure 11-13 STOP Power Mode (No STOP Bit Set)

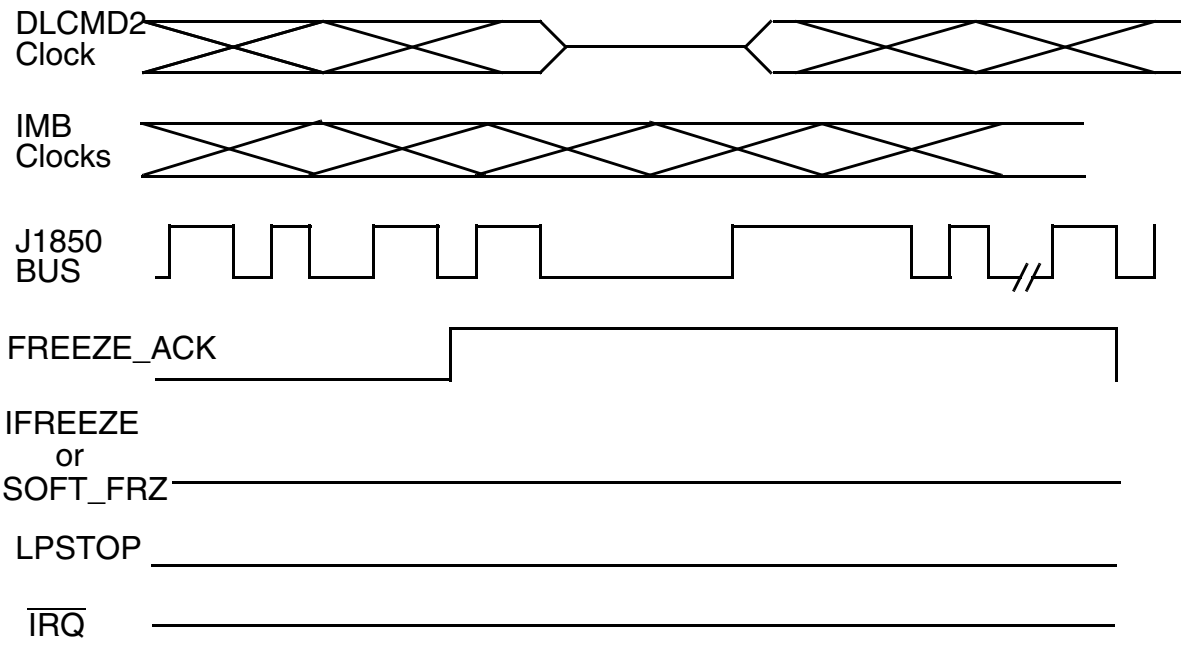


Figure 11-14 DEBUG Power Mode (IFEEZEB or SOFT_FRZ)

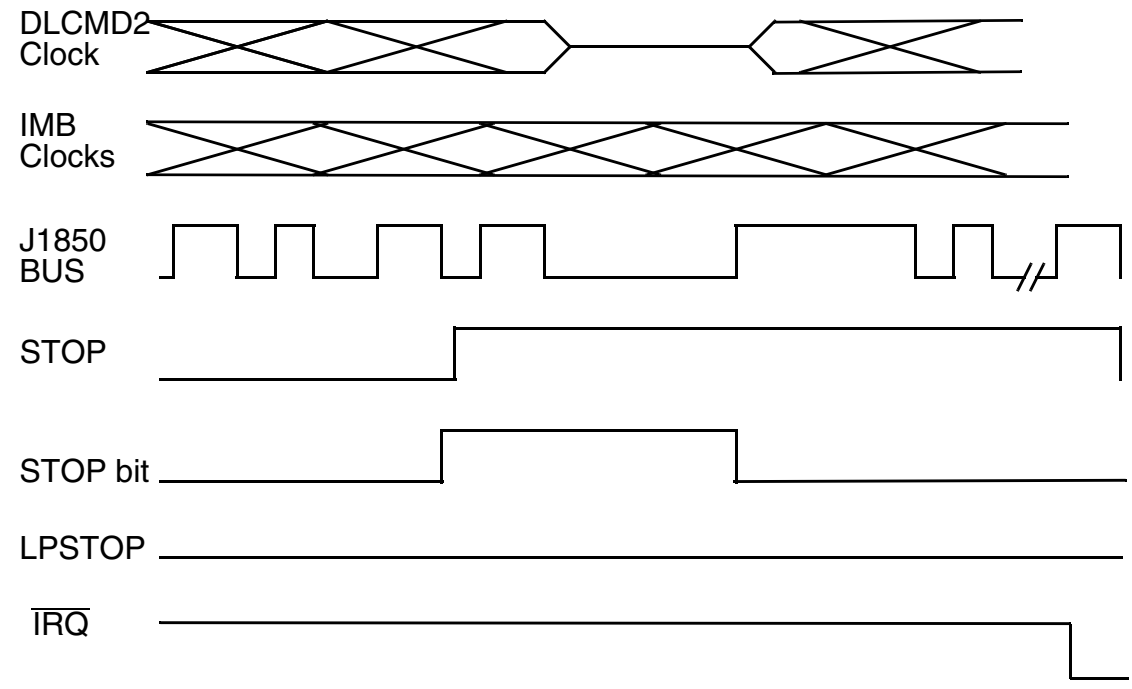


Figure 11-15 STOP Power Mode (STOP Bit Set)

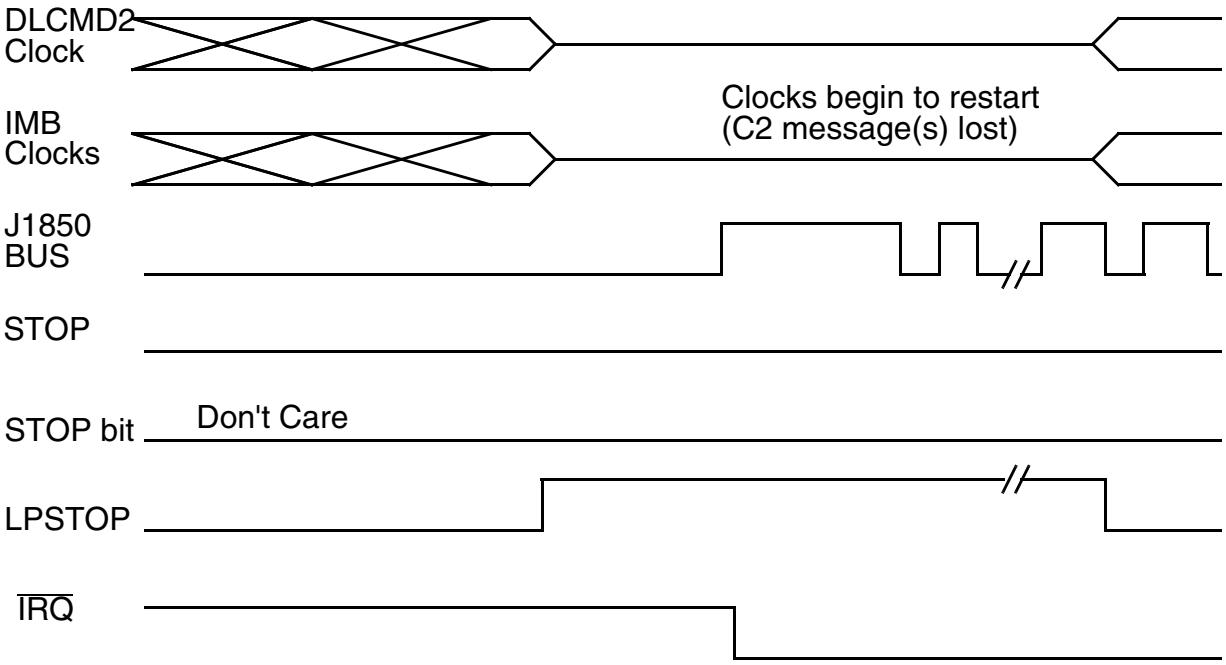


Figure 11-16 LPSTOP Power Mode

11.9 CPU Interface

This sections covers DLCMD2 interfaces to the CPU.

11.9.1 Interface Requirements

This section defines the interface protocol used.

The logical operations done in a DLCMD2 module service routine are as follows:

1. Read status byte from DLCMD2 module
2. Read received data from DLCMD2 module if required
3. Write command byte to DLCMD2 module if required
4. Write transmit data to DLCMD2 module

These four bytes are a complete exchange of information. However, these bytes are not all necessary or required during a CPU/DLCMD2 data link controller module (DLCMD2) module transfer. All possible unique CPU/DLCMD2 module transfers are shown in **Table 11-4**. All CPU/DLCMD2 module transfers will be made up of combinations of one or more of these building blocks. Pop and push refer to automatic (without command byte) flushing and loading of data bytes out of and into the DLCMD2 module.

A transmit data byte need not immediately follow the command. The transmit data byte must be the next byte transferred to the DLCMD2 but could be sent any time later. The command will not be acted upon until this next byte is sent to the DLCMD2.

Table 11-4 CPU/DLCMD2 Transfers

Acceptable Read/Write Combinations	Auto Pop?	Auto Push?	First Byte Flag Set?	Word Read/ Write Allowed?
1. Read status byte from DLCMD2[1]	No	No	No	No
2. Write command byte to DLCMD2	—	No	No	Yes
Write transmit or dummy data to DLCMD2 module	—	No	Maybe	Yes
If no data accompanies, then command byte causes no action	—	—	—	—
3. Read status byte from DLCMD2	No	—	—	Yes
Read received data byte from DLCMD2	Yes	—	—	Yes
Read status byte from DLCMD2	No	—	—	Yes
Read received data byte from DLCMD2	Yes	—	—	—
(Until entire message received)	—	—	—	—
4. Write transmit data byte to DLCMD2	—	Yes[2]	Yes	No
Write transmit data byte to DLCMD2	—	Yes	No	No
Write command byte to DLCMD2 (load as last byte)	—	No	No	Yes
Write transmit data to DLCMD2 (last byte)	—	No	No	Yes

NOTES:
1. Minimal transfer.
2. If TxFIFO empty.

In numbers 2-4, word reads and writes may be done to read a status byte and a received data byte, or write a command byte and a transmit data byte with one instruction. If number 2 was for loading a single byte for transmit the command byte would specify load as first and last byte (no auto push, no first byte flag set). Number 3 would be for reading an entire message from the DLCMD2 module when there is no transmit data to be sent to the DLCMD2 module. Number 4 is a quick way to load an entire message into the DLCMD2 module when there is no data to read from the DLCMD2 module.

Auto pop is the default for reading from the RxFIFO. This means that a read of a data byte from the RxFIFO causes the current byte in the FIFO to be automatically flushed. Auto pop can not be disabled. Auto push is the automatic loading of a data byte into the TxFIFO via a write to the DLCMD2 module. This is the default when there was no preceding command byte. There is no auto push if a command byte is sent before the data byte; and the command byte must specify what to do with the following data byte.

The action(s) called out in the command byte will be acted upon the moment the transmit (or dummy) data byte is written into the DLCMD2 module.

11.9.2 Reset Operation

When master reset is asserted, the DLCMD2 module will be held in an off state. System power should be up and stable when master reset is negated.

After toggling the reset line, the CPU writes command and any configuration bits into the DLCMD2 module to initialize it.

11.10 Operational Information

The following sections will be included in the DLCMD2 application document and are mostly redundant information to previous sections.

11.10.1 Initialization

After power up and/or reset the DLCMD2 should be configured via writes to the MCR, ILR,IVR, SDATA, and SCTL registers.

NOTE

Interrupts are disabled at power up and reset and must be enabled if a polled method of servicing the DLCMD2 is not used.

- Step 1 — Initialize MCR
  Begin initialization of the configuration bits by writing the SUPV bit in the MCR. This will determine what types of accesses are to be allowed (user and supervisor or just supervisor) to DLCMD2 registers for the rest of the initialization process and during normal operation. Care should be taken not to clear the SOFT_FRZ or FRZ[1:0] bits while writing the SUPV bit. This precaution will ensure the DLCMD2 remains in the debug mode until initialization is complete.
- Step 2 — Initialize ILR and IVR registers if interrupts employed
- Step 3 — Initialize SCTL and SDATA registers
- Step 4 — Enable DLCMD2 by exiting DEBUG mode

11.10.2 Transmitting a Message

The DLCMD2 is loaded with a message from the CPU for transmission. The DLCMD2 will then add a SOF bit to the outgoing data, and contend for a message slot on the J1850 bus. The TxFIFO in the DLCMD2 is 11 bytes long, to allow complete (except block) messages to be transferred to the DLCMD2 by the CPU for transmission. Information about which byte the CPU is transferring is sent to the DLCMD2 as part of the control information sent with each byte transferred and is optional with the DLCMD2 except for the last byte. When the DLCMD2 has transmitted all of the message bytes, it will automatically append a CRC to the end of the outgoing message.

The DLCMD2 will automatically retry a transmission if it lost arbitration or any errors were detected. The auto retry feature causes the DLCMD2 to indicate that the TxFIFO is full until the message is successfully sent except where no last byte was indicated. As soon as the CPU transfers a “last byte” of a message to the DLCMD2, or fills the eleventh position of the TxFIFO, the DLCMD2 will indicate that the TxFIFO is full. In the case of filling the TxFIFO with no “last byte” indicated (block mode) the status will say “TxFIFO almost full” after a byte has been sent so that the Data Link Controller Module (DLCMD2) next byte of the block message can be loaded by the CPU. Once successfully sent, the DLCMD2 will indicate that the TxFIFO is empty and hence ready for the next message. Each time the message loses arbitration, the completion code

for that received message will indicate to the transmitter that it attempted transmission, and lost arbitration. The auto retry feature can be terminated by the CPU through the command byte. This causes the DLCMD2 to finish its current transmit activity, and then clear the TxFIFO. If there had not been any transmit activity when the auto retry is disabled, the DLCMD2 will attempt to transmit the message once and then clear the TxFIFO.

A break waveform on the bus is shaped such that it will win arbitration against any currently transmitting message. When the DLCMD2 senses that such a waveform has occurred on the bus, it will stop transmitting its current message, reset its transmitter (clear the TxFIFO), and set a bit in the completion code and assert an interrupt to indicate to the CPU that such a condition has occurred. Since the break signal wins arbitration all of the time, any in progress messages will have simply lost arbitration, and the DLCMD2 will treat the in progress received message as complete. If a break occurs while there are no in progress messages, a completion code indicating that a break has occurred will be placed in the RxFIFO and a CPU interrupt generated. The break/reset waveform is sent by a DLCMD2 by setting a bit combination in the command byte. This waveform is sent immediately upon the command byte being latched in. If there is a current transmission, it will be stopped, and the break waveform sent. Break automatically takes all nodes on the network out of 4X speed mode.

NOTE

An SOF from a node in regular mode will be seen as a break by any nodes in 4X mode.

11.10.3 Receiving a Message

The status byte that the CPU reads from the DLCMD2 contains information on the status of the RxFIFO, the status of the TxFIFO, the condition of the bus, and the type of accompanying data byte. The data accompanying the status byte can be data received off of the J1850 busa completion code which contains information about a received message, or nothing.

All messages received off of the J1850 bus will have their start bit removed and error detection code replaced with a completion code. All other bytes of the message are placed in order, MSB first, in the RxFIFO.

Receiving information off of the J1850 bus occurs in much the same manner as sending data.

NOTE

By definition, every message a DLCMD2 sends on the data link is also received by its own receiver as if the transmission had been initiated by a different node.

The DLCMD2 will automatically remove the SOF bit from the message, and check the CRC for errors. If a CRC error occurs, it is flagged in the message completion infor-

mation (completion code) that takes the place of the CRC byte in the RxFIFO. The DLCMD2 will interrupt the CPU to signal that a complete message has been received, or when the RxFIFO is approaching full. When the CPU starts the transfer process, the status byte from the DLCMD2 indicates the data's presence, and something about the amount of data left in the RxFIFO. Conditions necessary for interrupting the CPU are selectable with a write to the interrupt level register (ILR). When a DLCMD2 loses arbitration to another node, it will continue to receive the remainder of the message. The completion code will indicate that the DLCMD2 contended for a transmit slot, and lost arbitration to the received message. The CPU does not need to resend the message if the auto retry feature is enabled.

11.10.4 Receiving a Message in Block Mode

Although not a part of the SAE J1850 protocol, the DLCMD2 does allow for a special "block mode" of operation for the receiver. As far as the DLCMD2 is concerned, a block mode message is simply a long J1850 frame that contains an indefinite number of data bytes. All of the other features of the frame remain the same, including the SOF, CRC, and EOD symbols.

Data link controller module (DLCMD2) class 2 convention requires that another node wishing to send a block mode transmission must first inform all other nodes on the network that this is about to happen. This is usually accomplished by sending a special predefined message.

Block mode may be combined with 4X mode.

11.10.5 Transmitting a Message in Block Mode

If the TxFIFO was filled with no last byte indicated, the DLCMD2 will start sending that message in terminate auto-retry mode; the status will indicate "TxFIFO almost full" as soon as the first byte is sent, and "TxFIFO contains some data bytes" as soon as the second byte is sent. As new data is written to the TxFIFO, the status will accurately reflect the condition of the TxFIFO until a "last byte" is written. When a "last byte" command has been sent to the DLCMD2, the TxFIFO will indicate "TxFIFO full" until the transmission is finished. The DLCMD2 will send an infinite length message if properly handled by the CPU.

Block mode transmit may be combined with 4X mode.

11.10.6 Receiving a Message in 4X Mode

Although not a part of the SAE J1850 protocol, the DLCMD2 does allow for a special "4x mode" of operation for the receiver. 4X mode is entered by the programmer setting the 4X bit in the MCR register. This bit is cleared automatically by a BREAK symbol reception, or may be manually cleared by the programmer.

11.10.7 Transmitting a Message in 4x Mode

If the programmer sets the 4X mode bit in the MCR register, the DLCMD2 will use a different set of VPW timing numbers to set the widths of the J1850 bus symbols when

transmitting. This bit is cleared automatically by a BREAK symbol reception, or it may be cleared manually by the programmer.

11.11 Register Descriptions

The following section provides register descriptions for the DLCMD2. In the following paragraphs, the top line lists the bit number in the register. The second line contains the mnemonic for the bit. The values shown under the mnemonic are the values of those register bits after IMSTRSTB is asserted. If ISYSRSTB affects the bits differently than IMSTRSTB, this fact is discussed in the paragraphs following the chart.

The DLCMD2 registers hold, in some cases, bit locations marked as “reserved”. These bits will always read as logic ‘0’ and writes to these bits are ignored.

The register decode map is fixed and begins at the first address of the module base address. **Table 11-5** shows the registers associated with the DLCMD2 module and their relative offset from the base address. Four of the registers are in supervisor-only data space and the remainder are in assignable data space.

Table 11-5 DLCMD2 Memory Map

Access	Offset	15	0	R/W	Reset
S	0xYF F600	Module Configuration Register (MCR)		R/W	0x6780
S	0xYF F602	Reserved		—	—
S	0xYF F604	Interrupt Pending Register (IPR)		R/W	0x0000
S	0xYF F606/ 0XYF F607	Interupt Level Register (ILR)	Interrupt Vector Register (IVR)	R/W, RO (bit2~bit0)	0x000f(IRQ_PLUG=0) 0x000f(IRQ_PLUG=1)
S/U	0xYF F608	Symbol Timing Control and Pre-scaler Register (SCTL)		R/W	0x0000
S/U	0xYF F60A	Symbol Timing Data Register (SDATA)		WO	0x00xx
S/U	0xYF F60C	Command Register (CMD)	Transmit Data Register (TDATA)	R/W-WO for (TDATA)	0x00xx
S/U	0xYF F60E/ 0XYF F60D	Status Register (STAT)	Receive Data Register (RDATA)	RO	0x00xx

11.11.1 Module Configuration Register (MCR)

MCR— Module Configuration Register **0xYF F600**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
STOP	FRZ		DSAE	4XMD	SOFT_FRZ	NOT_RDY	FREEZ_ACK	SUPV	0	0	STOP_ACK	IARB			
RESET:															
0			0	0	1	1	1	1	0	0	0	0	0	0	0

= Unimplemented

Table 11-6 MCR Bit Descriptions

Bit(s)	Name	Description
15	STOP	Stop system clock. The STOP bit, if asserted, will stop the system clock within the DLCMD2 module except for the IMB bus interface (BIU). The module's BIU must continue to operate to allow the CPU access to the module's registers (except for LPSTOP). The system clock is stopped on the low phase of ICLOCK. Once the bus becomes idle, the system clock will remain stopped until the STOP bit is negated by the CPU, or a reset occurs, or an edge from the J1850 bus passes through the digital filter (if DSAE not set).
14:13	FRZ	Freeze bit. There exist two bits in the DLCMD2 module control register to determine the action to be taken when the freeze signal of the IMB is asserted. **Table 11-7** defines the freeze bit field
12	DSAE	Disable STOP mode automatic exit. When asserted, the DSAE bit will prevent J1850 activity from causing the DLCMD2 to exit STOP mode, and will prevent INTACL2-type interrupts from being asserted. (When DSAE is negated, the DLCMD2 will automatically restart its internal clocks from STOP mode upon sensing any J1850 activity, and INTACL2-type interrupts will be allowed (although they must still be explicitly enabled by INTACL2E).) The CPU must negate the STOP bit to exit DLCMD2 STOP mode and restart the DLCMD2 clocks, since J1850 activity will not take the DLCMD2 out of STOP mode. When cleared, any J1850 activity that passes through the digital filter will take the DLCMD2 out of STOP mode and clear the STOP bit (if set).
11	4XMD	4X mode. When the 4XMD bit is asserted, the DLCMD2 will use 4X mode bit timings rather than the normal J1850 symbol timings. Note that normal waveshaping by the analog transceiver (whether on-chip or off-chip) must be disabled for the DLCMD2/transceiver combination to transmit properly in 4X mode. This bit is automatically reset upon receipt of a BREAK symbol.
10	SOFT_ FRZ	Software freeze. Assertion of this bit has the same effect as the assertion of the IFREEZEB signal on the IMB3, as described in the description of IFREEZEB/SOFT_FRZ mode and FRZE bits. However, it does not require that the IFREEZBE signal be asserted in order to enter debug mode. This bit is initialized to 1 (debug mode).The CPU clears it after initializing the control registers. No transmissions or receptions are performed by the DLCMD2 before this bit is cleared. For detailed description of the debug mode, refer to **11.8.5 DLCMD2 DEBUG**. 0 = No DLCMD2 internal request to enter Debug Mode. 1 = Enter Debug mode if (FRZE[1] || FRZE[0])
9	NOT_ RDY	Not ready. This bit indicates that the DLCMD2 is either in STOP or in DEBUG state. This bit is read-only. Whenever one of these two modes is asserted, this bit is set once the DLCMD2 has entered the corresponding mode. It is negated once the DLCMD2 has exited these modes. For more details refer to **11.8.4 DLCMD2 STOP and LPSTOP** and LPSTOP and **11.8.5 DLCMD2 DEBUG**.
8	FREEZ_ ACK	DLCMD2 disabled. DLCMD2 is in debug mode and its pre-scaler is stopped. This bit is read-only. When the DLCMD2 enters debug mode it sets the FREEZ_ACK bit. The CPU can poll this bit to know if the DLCMD2 entered debug mode. If debug mode is negated then this bit is negated once the DLCMD2's pre-scaler is running. 0 = Debug mode is negated and the pre-scaler is running again 1 = The DLCMD2's pre-scaler is stopped
7	SUPV	Supervisor/user data space. The SUPV bit affects the SCTL/SDATA, CMD/TDATA and STAT/RDATA registers, the only registers in the DLCMD2 currently defined as supervisor/unrestricted access. If SUPV is asserted (all DLCMD2 registers are to be treated as supervisor only) bit 2 of the function code (FC2) must be asserted during module address decoding to allow the supervisor/unrestricted access registers to respond to accesses in supervisor data space. If the SUPV bit is cleared (SCTL/SDATA, CMD/TDATA and STAT/RDATA accesses are to be treated as unrestricted) FC2 is ignored during module address decoding, allowing supervisor/unrestricted access registers to respond to accesses in supervisor data or user data space.
6:5	—	Reserved

Table 11-6 MCR Bit Descriptions (Continued)

Bit(s)	Name	Description
4	STOP_ ACK	STOP acknowledge. DLCMD2 is in STOP mode and its main clocks are stopped, (see **11.8.4 DLCMD2 STOP and LPSTOP**) This bit is read-only. When the DLCMD2 enters STOP mode and shuts its clocks off, it sets the STOP_ACK bit. The CPU can poll this bit to know if the DLCMD2 entered stop mode (e.g., stopped its clocks). If stop mode bit is negated, then this bit is negated once the DLCMD2's clocks are running. 0 = STOP mode is negated and the DLCMD2's clocks are running again 1 = The DLCMD2 enters STOP Mode and its clocks are stopped
3:0	IARB	Interrupt arbitration. This four-bit encoded field contains the interrupt arbitration number of this particular DLCMD2 module with respect to all other subsystems and peripherals that may generate interrupts. The interrupt arbitration ID is used to arbitrate the IMB3 (relevant only when IRQ_PLUG = 0) when two or module/peripherals have an interrupt on the same priority level pending simultaneously. This field is initialized to the non-arbitrating state, $0 during reset. If no arbitration takes place during the IACK cycle, the spurious interrupt vector is generated after a time-out by the SIM module, alerting the system that an interrupt arbitration ID has not been initialized. The IARB field should be initialized by the system software to a value between 0xF (highest priority) and 0x1 (lowest priority).

Table 11-7 Freeze Bit Field Description

FRZ1	FRZ0	Result
0	0	Ignore FREEZE
0	1	Freeze on DLCMD2 internal f_{DLC} high state. (CTWO is in high state when IFREEZEB is asserted, or f_{DLC} enters its high state sometime after IFREEZEB is asserted.)
1	0	Freeze on receipt of next bit. (DLCMD2 internal RIT signal is asserted IFREEZEB is asserted, or RBIT becomes asserted sometime after IFREEZEB is asserted.)
1	1	Freeze immediately

11.11.2 Interrupt Pending Register (IPR)

IPR — Interrupt Pending Register **0xYF F604**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
0	0	0	0	0	0	0	0	0	0	0	ipr_4	ipr_3	ipr_2	ipr_1	ipr_0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

= Unimplemented

Table 11-8 IPR Bit Descriptions

Bit(s)	Name	Description
15:5	—	Reserved
4:0	IPR[4:0]	Interrupt pending. The interrupt pending register (IPR) indicates that an interrupt service request has been made by the module interrupt logic. An interrupt request to the IMB3 is generated whenever the interrupt asserting condition is met. A bit in the IP register (indicating the application logic interrupt request is detected) can be set in any order to generate the IMB3 interrupt request. See **Figure 11-17** for the interrupt request logic path. Once set, the IP bits remain set until the IP bit is cleared by software, or reset. To clear an IP bit, the bit must be first read as a 1 and then the bit must be written to a 0. IP bits which are 0 when the IP register is read are unaffected by the write operation. Also, if the IP logic detects another application logic interrupt request after the IP bit was read as a 1 and before a 0 is written to clear it, the IP bit cannot be cleared until the IP bit is again read as a 1 and then written to a 0. Bits in this register are set by the application logic request and cannot be written to a one by software (writing 1 to the IP register have no effect). Only writes of 0 are valid, when permitted, to clear the IP bit(s). IP_0 : R1STBYTE interrupt pending setsIP[0]. IP_1 : RCCODE interrupt pending sets IP[1]. IP_2 : R13BYTE interrupt pending sets IP[2]. IP_3 : THLFMTY interrupt pending sets IP[3]. IP_4 : ACL2 interrupt pending sets IP[4].

11.11.3 Interrupt Level Register (ILR)

ILR — Interrupt Level Register **0xYF F606**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
INTMODE	INTACL2E	RESERVED			ILR			INTERRUPT VECTOR REGISTER (IVR)							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

= Unimplemented

Table 11-9 ILR Bit Descriptions

Bit(s)	Name	Description
15	INTMODE	Interrupt mode. When the INTMODE bit is asserted, the DLCMD2 will assert an interrupt when a single byte is received into an empty RxFIFO. When cleared, only standard interrupts are enabled.
14	INTACL2E	Interrupt any bus activity enable. When the INTACL2E bit is set, the DLCMD2 will assert an interrupt when any network activity (including noise) is detected. This bit must be set for the DLCMD2 to wake up the processor from LPSTOP. Although the detected activity may be only noise, neither IMB3 nor module clocks can run in LPSTOP, therefore the DLCMD2 cannot distinguish noise from valid network activity, and must wake up the processor to restart the clocks necessary for J1850 message reception. Normally, this bit would be set by the CPU just before going into LPSTOP, and is cleared once the ACL2 interrupt condition is latched.
13:11	—	Reserved

Table 11-9 ILR Bit Descriptions (Continued)

Bit(s)	Name	Description
10:8	ILR	Interrupt request level. The interrupt request level field contains the priority level of the DLCMD2 interrupts for the CPU. Level seven for this field indicates a nonmaskable interrupt, while level zero indicates that interrupts have been disabled. The interrupt request level field is initialized to zero during reset which prevents the module from generating an interrupt until this register has been initialized. Note: level zero corresponds to IRQ[0] which is not recognized at the system level, hence the interrupts are treated as disabled. The interrupt request level field, therefore acts as master enable for the interrupts.
7:0	IVR	Interrupt vector register. The interrupt vector register holds the offset into the exception vector table, and is what is driven by the DLCMD2 in response to an IMB IACK cycle. IVR[7:3] are programmable by the user. IVR are read-only bits and encoded from the highest priority of any currently active interrupt sources per Table 11-34. All reads of the IVR register will return $0F, the defined state for an uninitialized interrupt vector, until a write access to this register occurs. Therefore IVR[2:0] would not be updated from DLCMD2's internal interrupt status until IVR is written once by the software when IRQ_PLUG =0..

Table 11-10 IVR Encoding

ACL2[1]	THLFMTY[2]	R13BYTE[3]	RCCODE[4]	R1STBYTE[5]	IVR2	IVR1	IVR0
A[6]	x	x	x	x	1	0	1
N[7]	A	x	x	x	1	0	0
N	N	A	x	x	0	1	1
N	N	N	A	x	0	1	0
N	N	N	N	A	0	0	1
N	N	N	N	N	0	0	0

NOTES:
1. ACL2 = Any J1850 activity, DSAE bit in MCR register is cleared and INTACL2E in ILR is set.
2. THLFMTY = TxFIFO half empty.
3. R13BYTE = RxFIFO has 13 bytes and no completion code.
4. RCCODE = RxFIFO has a complete message.
5. R1STBYTE = RxFIFO has one data byte and nothing else.
6. A = Assert interrupt.
7. N =No interrupt for this event.

11.11.4 Symbol Timing Control and Pre-Scaler Register (SCTL)

SCTL — Symbol Timing Control and Pre-Scaler Register **0xYF F608**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
0	NBFS	RXPOL	0	0	0	LCK	SEL	0	0	PS5	PS4	PS3	PS2	PS1	PS0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

= Unimplemented

Table 11-11 SCTL Bit Descriptions

Bit(s)	Name	Description
15	—	Reserved
14	NBFS	Normalization bit (NB) format. This bit controls the format of normalization bit (NB). SAE J1850 strongly encourages the use of an active long, ‘0’, for in-frame responses containing CRC and active short, ‘1’, for in-frame response without CRC. Once the LCK is set, the writes to NBFS are disabled. 0 = NBFS, NB that is received or transmitted is a ‘1’ when the response part of an in-frame response (IFR) ends with a CRC byte. NB that is received or transmitted is a ‘0’ when the response part of an in-frame response (IFR) does not end with a CRC byte. 1 = NBFS, NB that is received or transmitted is a ‘0’ when the response part of an in-frame response (IFR) ends with a CRC byte. NB that is received or transmitted is a ‘1’ when the response part of an in-frame response (IFR) does not end with a CRC byte
13	RXPOL	Receive pin polarity. The receive pin polarity bit is used to select the polarity of incoming signal on the receive pin. Some external analog transceivers invert the receive signal from the J1850 bus before feeding back to the digital receive pin. Once the LCK is set, the writes to RXPOL are disabled. 0 = RXPOL, select normal/true polarity; true non-inverted signal from J1850 bus, (i.e., the external transceiver does not invert the receive signal). 1 = RXPOL, select inverted polarity, where external transceiver inverts the receive signal
12:10	—	Reserved
9	LCK	Write lock on symbol timing parameter table. LCK disables writes to the symbol timing parameter table through SDATA. Once LCK is set, only reset will clear the bit (unless in test mode). 0 = LCK, enables the writes to the symbol timing parameter table through SDATA register. 1 = LCK, disables the writes to SEL, NBFS, RXPOL, PS5 -PS0 in SCTL register and the symbol timing parameter table through SDATA register. Once LCK is set, only reset will clear this bit (unless in test mode).
8	SEL	Select normal or 4X bit-timing parameter table. There are two sets of parameters used for DLMCD2 bit-timing. These two sets of parameters are accessible through the SDATA register. There are twelve parameters for normal bit-timing (four for receiving and eight for transmitting) and twelve parameters for 4X bit-timing (four for receiving and eight for transmitting). These bits may be written only when LCK is cleared. Once the LCK is set, the writes to SEL are disabled and SEL is cleared. 0 = SEL, allows access (write) to the normal mode parameters. 1 = SEL, allows access (write) to the 4X mode parameters
7:6	—	Reserved
5:0	PS[5:0]	Pre-scaler rate select. Set the system clock divisor necessary to achieve the DLCMD2 internal bit-rate clock. The frequency should be as close to 2 MHz as possible. These bits may be written only when LCK is cleared. Once the LCK is set, the writes to PS5-PS0 are disabled. The value programmed into PS5-PS0 bits is dependent on the chosen IMB3 system clock frequency per **Table 11-12**. Note: The PS5-PS0 is always loaded with “desired divisor” -1 and comes out of reset programmed for a divided by one clock rate (PS5-PS0 = 0x00)

Table 11-12 DLCMD2 Pre-Scaler Rate Selection

IMB3 Bus Clock Frequency	PS5-PS0	Division	f_{DLC}
f_{CLOCK} = 2.000 MHz	0x00	1	2.000 MHz
f_{CLOCK} = 15.000 MHz	0x06	7	2.143 MHz
f_{CLOCK} = 16.000 MHz	0x07	8	2.0 MHz
f_{CLOCK} = 25.000 MHz	0x0B	12	2.083 MHz
f_{CLOCK} = 26.000 MHz	0x0C	13	2.0 MHz
f_{CLOCK} = 40.000 MHz	0x13	20	2.0 MHz
f_{CLOCK} = 45.000 MHz	0x15	22	2.045 MHz
f_{CLOCK} = 66.000 MHz	0x20	33	2.0 MHz
f_{CLOCK} = 128.000 MHz	0X3F	64	2.0 MHz

11.11.5 Symbol Timing Data Register (SDATA)

SDATA — Symbol Timing Data Register **0xYF F60A**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
0	0	0	0	0	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

= Unimplemented

Table 11-13 SDATA Bit Descriptions

Bit(s)	Name	Description
15:11	—	Reserved
10:0	S[10:0]	Symbol timing data. The bit-timing of J1850 symbols is written into the twenty-four entries of the parameter table through the SDATA register. An internal pointer along with the SEL bit is used to select which parameter is accessed (write) through the SDATA register. This pointer is incremented when SDATA is written and cleared when the SCTL register is accessed. Refer to **Figure 11-17** for a block diagram and **Table 11-14** for the parameter table. The parameter table lists the cycle counts for four receive symbols and eight transmit symbols in normal and 4X modes. The user must calculate the cycle counts for each symbol based on the desired value, round trip delay, digital filter delay, and bus frequency. The calculated cycle counts must be entered into the parameter table through the SDATA. To calculate the cycle count for receive symbols, multiply f_{DLC} by the desired symbol time (in µs) and round to the nearest integer. To calculate the cycle count for transmit symbols, subtract the round trip delay for the transceiver from the desired symbol time (in µs), multiply by f_{DLC}, round to the nearest integer, and subtract the cycle count of the digital filter (16 cycles in normal mode, four cycles in 4X mode). As an example, let the transceiver round trip delay be 16 µs, the desired symbol be 64 µs, and fDLC be 2.083 MHz. The cycle count would be: cycle count = ((64 µs - 16 µs) x 2.083 cycles / µs) - 16 cycles = 84 cycles After the cycle count has been computed, it must be converted into binary format and entered into the respective parameter through the SDATA.

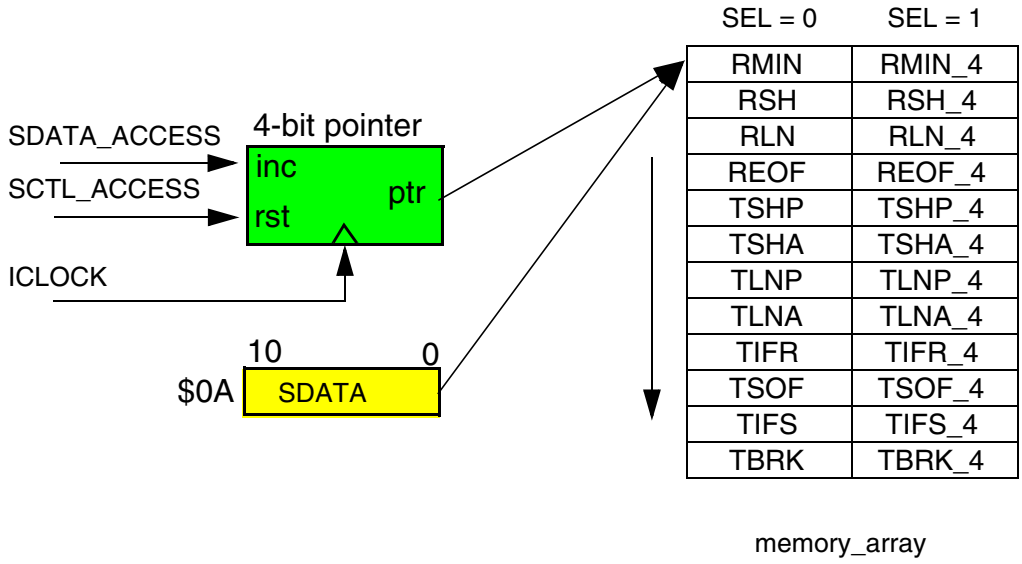


Figure 11-17 DLCMD2 SDATA Block Diagram

Table 11-14 Timing Parameter Table

Programming Sequence	Symbol Description	Symbol (SEL = 0)	Symbol Timing (µS)	Symbol (SEL = 1)	Symbol Timing 4X (µs)
1	Minimum receive symbol time	RMIN	34	RMIN_4	8
2	Maximum receive short pulse time	RSH	96	RSH_4	24
3	Maximum receive long pulse time	RLN	163	RLN_4	41
4	Minimum end-of-frame time	REOF	239	REOF_4	60
5	Target transmit short passive symbol time	TSHP	64	TSHP_4	16
6	Target transmit short active symbol time	TSHA	64	TSHA_4	16
7	Target transmit long passive symbol time	TLNP	128	TLNP_4	32
8	Target transmit long active symbol time	TLNA	128	TLNA_4	32
9	Target end-of-data time before IFR begins	TIFR	200	TIFR_4	50
10	Target transmit start-of-frame symbol time	TSOF	200	TSOF_4	50
11	Target inter-frame separation time before DLCMD2 is ready for new message	TIFS	320	TIFS_4	80
12	Target transmit break symbol time	TBRK	800	TBRK_4	800

11.11.6 Command Register (CMD)

Command and configuration of the DLCMD2 module is accomplished through a command byte that accompanies every data byte sent from the CPU to the DLCMD2 module.

NOTE

The command byte is the first byte transferred to the DLCMD2 module, and must be followed by a byte (data or dummy) which causes the command to be acted on (this can be done in an aligned word write or two separate byte writes).

CMD — Command Register **0xYF F60C**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CMD								Transmit Data Register (TDATA)							
RESET:															
0	0	0	0	0	0	0	0								

Table 11-15 CMD Bit Descriptions

Bit(s)	Name	Description
15:8	CMD	Command byte. The command byte can be broken in three fields. Bits 7:5 describe general commands. Bits 4:3 describe the type and destination of the accompanying byte associated with the command. Bit 1:0 control the receive FIFO. Refer to **Table 11-16** for general command descriptions. Refer to **Table 11-17** for type and destination of accompanying byte descriptions. Refer to **Table 11-19** for RxFIFO command descriptions.
7:0	TDATA	Transmit data register. See **11.11.7 Transmit Data Register (TDATA)**.

Table 11-16 General Commands

CMD 7	CMD6	CMD5	Description
0	0	0	Do nothing — DLCMD2 will not perform any actions defined in this field.
0	0	1	Reserved — Results in “do nothing.”
0	1	0	Send Break signal — When this command is latched in, the DLCMD2 will immediately send a break signal on the bus, regardless of its current transmit or receive status.

Table 11-16 General Commands (Continued)

CMD 7	CMD6	CMD5	Description
0	1	1	Send on EOD with CRC (Transmit Single or Multiple Byte IFR with CRC) 1. Must be accompanied by a data byte that is flagged as “1st byte” or “1st and last” byte. The remaining IFR bytes may not be sent to the DLCMD2 with the “Send on EOD” combination set. 2. This command causes the DLCMD2 to go into Terminate Auto Retry (TAR) mode automatically. It also causes a reset of the transmitter and TxFIFO before the response byte is loaded. If the DLCMD2 is transmitting and loses arbitration, an IFR can be loaded in response to the message winning arbitration. 3. If this command is latched in while the bus is idle or between EOD and EOF, the IFR command/data bytes will be ignored. If valid command/data bytes follow the invalid IFR command/data bytes, the DLCMD2 will attempt to transmit the bytes as a normal message once. 4. If a message is in progress and the EOD symbol has not been detected when this command is latched in, the DLCMD2 will send the IFR if there were no receiver errors detected. If receiver errors were detected, the IFR will be lost and the DLCMD2 will not be in TAR mode. 5. If the remaining IFR bytes, if any are required, are not placed into the TxFIFO before they are needed, then an underrun will occur. This will cause the CRC to be inverted in the IFR. The second byte must be loaded before the falling edge of the Normalization Bit. The next IFR byte must be loaded before the last bit of the current byte is transmitted. Since there is only one responder transmitting an IFR, the inverted CRC will cause the DLCMD2 to corrupt its own message. Other nodes will receive this as a CRC error. 6. If this DLCMD2 is the one transmitting, send on EOD will cause a bit timing or incomplete byte error and a reset of the TxFIFO. Both the original message and the IFR will be lost. 7. Cannot be sent in response to an IFR without CRC.
1	0	0	Terminate Auto Retry (TAR) 1. If this command is latched in while there is not a complete message in the TxFIFO, then when that message is completely loaded the DLCMD2 will attempt to transmit that message only once. After this transmit attempt, the TxFIFO will be empty. 2. If this command is latched in while the DLC is transmitting, then the DLCMD2 will finish its transmit activity (successful transmission, or lose arbitration) and then clear the TxFIFO. 3. If a complete message is in the TxFIFO, but not yet transmitting when this command is latched in, and: a. The message has just lost arbitration and is waiting for the next slot for retransmission, the DLCMD2 will attempt transmission one more time. b. The message has not tried to transmit yet (loaded while a message was on the bus). The DLCMD2 will try to transmit once, and then reset the TxFIFO. c. If the TxFIFO is full, and there is no last byte indicated (message of more than 11 bytes), then an automatic TAR is executed.
1	0	1	Send on EOD without CRC (Transmit Single or Multiple Byte IFR w/o CRC) 1. Same as 0 1 1 but a CRC will not be transmitted. 2. This response cannot be sent after a previous IFR without CRC.

Table 11-16 General Commands (Continued)

CMD 7	CMD6	CMD5	Description
1	1	0	Send on EOD with auto re-try (Transmit Single Byte IFR with auto re-try) 1. If a loss of arbitration occurs when the DLCMD2 attempts to transmit and after the IFR byte winning arbitration completes transmission, the DLCMD2 will again attempt to transmit. The DLCMD2 will continue transmission attempts until an error is detected on the bus. 2. The TxFIFO will not be empty until the message is successfully sent.
1	1	1	Abort Transmission Now (Reset Transmitter) 1. This command causes the transmitter, including the TxFIFO to be reset immediately. If a message is in progress, it will be terminated immediately. 2. The first byte of a new message may accompany this command in the same 2-byte host-DLCMD2 transfer.

Table 11-17 Type and Destination of Accompanying Byte Commands

CMD 4	CMD 3	CMD 2	Description
0	0	0	Do Not Load as Transmit Data 1. The DLCMD2 will not perform any actions defined in this field.
0	0	1	Load as Transmit Data 1. If the TxFIFO is indicating full (complete message already in TxFIFO), then the byte is not loaded and is lost.
0	1	0	Reserved
0	1	1	Load as Last Byte of Transmit Data 1. If there is not a “1st byte” at the head of the TxFIFO, the byte is not loaded and is lost unless a block transfer is in progress. 2. If the TxFIFO is empty, the byte is not loaded and is lost. 3. If the TxFIFO is indicating full (complete message already in TxFIFO), then the byte is not loaded and is lost.
1	0	0	Reserved
1	0	1	Load as First Byte of Message 1. If there is already a first byte in the TxFIFO, the byte is loaded, and will cause a transmit underrun error during transmission of this message. The first “first byte” and subsequent bytes will transmit correctly until the second “first byte” is encountered and this byte is not transmitted and is lost. 2. If the status of the TxFIFO is full (complete message already loaded, or partial message < 11 bytes), then the data byte is not loaded, and is lost.
1	1	0	Reserved
1	1	1	Load as First and Last Byte of Message 1. If the TxFIFO is full, then the byte is not loaded and is lost.

11.11.7 Transmit Data Register (TDATA)

TDATA — Transmit Data Register **0xYF F60C**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CMD								Transmit Data Register (TDATA)							
RESET:								0	0	0	0	0	0	0	0

Table 11-18 TDATA Bit Descriptions

Bit(s)	Name	Description
15:8	CMD	Command byte. See **11.11.6 Command Register (CMD)**.
7:0	TDATA	Transmit data register. This register is used to load data into the TxFIFO. Data is automatically pushed into the TxFIFO on every write to the TDATA register. See **Figure 11-18**.

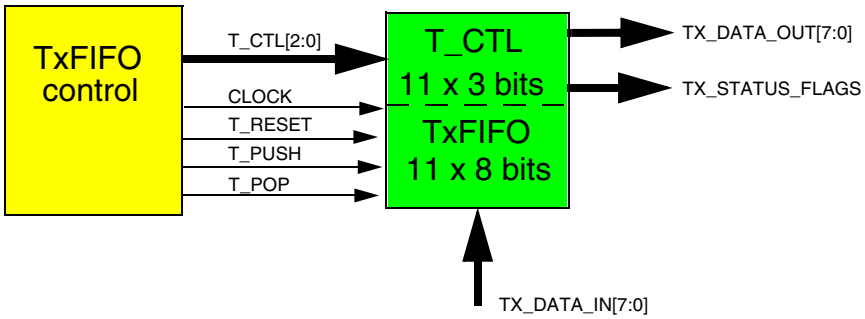


Figure 11-18 TxFIFO Block Diagram

Table 11-19 RxFIFO Commands

CMD 1	CMD 0	Description
0	0	Do Nothing
0	1	Reserved (Do nothing)
1	0	Flush Byte 1. If there is nothing in the RxFIFO, then no action is taken even if a byte arrives before the DLCMD2 access is complete. 2. If there are any bytes in the RxFIFO, then the first byte (at the head of the RxFIFO) is flushed. 3. Flush byte commands will not queue up.
1	1	Flush Current Message[1] 1. If there is nothing in the RxFIFO when this command is latched in, the next message received will be flushed except for the completion code. Only the completion code will cause an interrupt. 2. If there is a partial message in the RxFIFO (still being received off of the data link), all remaining bytes of the message will be flushed, except for the completion code, when it is formed. Only the completion code will cause an interrupt. 3. If there is a complete message in the RxFIFO (message bytes and completion code), all bytes of the message will be flushed except for the completion code. 4. If there is completion code at the head of the RxFIFO, no action is taken when this command is latched in.

NOTE:
1. Flush message commands cannot be queued up. Only one dump command is ever active. A Flush mes-sage command can not be stopped after being issued. In cases 2 and 3, the RxFIFO status may be invalid during a flush message operation. It could take a few þs (or even longer if there is no completion code in the RxFIFO yet) for the status to correctly indicate the condition of the RxFIFO. In cases 1 and 2, the RxFIFO status invalid time varies with how much of the message is yet to be received, (i.e., the completion code must “leak down” to the head of the RxFIFO before the RxFIFO is “valid” again). The RxFIFO is not necessarily empty after a flush message command.

11.11.8 TxFIFO Command Load Sequences

The following table shows various load sequences for the CMD and TDATA registers. The table indicates whether the sequence is valid or invalid and how the DLCMD2 will handle each case.

Table 11-20 Command Load Sequences

Load Sequence	DLCMD2 operation
1) Write “Load as First Byte” and data byte as 16-bit write. 2) Write “Load as Transmit Data Byte” and data byte as 16-bit write. 3) Write “Load as Last Byte” and data byte as 16-bit write.	All three command and data bytes are pushed into the TxFIFO. DLCMD2 transmits message correctly. “Load as Last Byte” command and last data byte can be written with two separate 8-bit writes.
1) Write data bytes as 8-bit writes. 2) Write “Load as Last Byte” and last data byte as 16-bit write.	First data byte treated as “first byte.” “Load as Last Byte” and last data byte is pushed into the TxFIFO. DLCMD2 transmits message correctly. “Load as Last Byte” command and last data byte can be written with two separate 8-bit writes.
1) Write “Send on EOD” as 8-bit write. 2) Write data bytes as 8-bit writes. 3) Write “Load as Last Byte” and last byte as 16-bit write.	No IFR transmission occurs. “Send on EOD” command will be ignored and first data byte will be treated as “first byte.” If EOF is received, the DLCMD2 will transmit the bytes as a normal message.
1) Write “Send on EOD” as 8-bit write. 2) Write “Load as First Byte” and data byte as 16-bit write. 3) Write data bytes as 8-bit writes. 4) Write “Load as Last Byte” and data byte as 16-bit write.	No IFR transmission occurs. “Send on EOD” command will be ignored and first data byte will be treated as “first byte.” If EOF is received, the DLCMD2 will transmit the bytes as a normal message.
1) Write “Send on EOD,” “Load as First Byte,” and data byte as 16-bit write. 2) Write data bytes as 8-bit writes. 3) Write “Load as Last Byte” and last data byte as 16-bit write.	Command and data word is pushed into the TxFIFO. Data bytes are pushed into the TxFIFO. Command and data byte is pushed into the TxFIFO. DLCMD2 transmits an IFR correctly.
1) Write “Load as First Byte” as 8-bit write. 2) Write “Send on EOD” as 8-bit write. 3) Write data bytes as 8-bit writes. 4) Write “Load as Last Byte” and data byte as 16-bit write.	No IFR transmission occurs. “Send on EOD” command will be ignored and first data byte will be treated as “first byte.” If EOF is received, the DLCMD2 will transmit the bytes as a normal message.
1) Write “Send on EOD,” “Load as First Byte,” and data byte as 16-bit write. 2) Write “Send on EOD,” “Load as transmit data byte,” and data byte as 16-bit write. 3) Write “Send on EOD,” Load as Last Byte,” and data byte as 16-bit write.	No IFR transmission occurs. Data bytes other than the first byte of an IFR cannot be written with “Send on EOD.” All data bytes are ignored.

NOTE

Any load sequence other than sequences 1-5 will not result in DLCMD2 transmission activity.

11.11.9 Status Register (STAT)

Each byte of data that the DLCMD2 module has received from the J1850 bus will be transferred to the CPU along with a status byte, which relays information on the condition of the DLCMD2 module and the data that has been received.

NOTE

Commands that are in progress may not be reflected in the status byte until sufficient time has elapsed for that command to be performed. The status must be read with each received data byte on the DLCMD2 module, either in an aligned word read, or two separate byte reads.

STAT — Status Register **0xYF F60E**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
STAT								Receive Data Register (RDATA)							
RESET:															
0	0	0	0	0	0	0	0								

Table 11-21 STAT Bit Descriptions

Bit(s)	Name	Description
15:8	STAT	Status byte. The status byte can be broken in five fields. Bits 7:5 describe the status of the RxFIFO. Bits 4 indicates whether the bus is idle. Bit 3 indicates whether the bus is shorted to ground. Bit 2 indicates whether the TxFIFO is underrunning. Bit 1:0 describes the status of the TxFIFO. Refer to **Table 11-22** for RxFIFO status descriptions. Refer to **Table 11-23** for data link idle status descriptions. Refer to **Table 11-24** for transmitter shorted status descriptions. Refer to **Table 11-25** for TxFIFO underrunning status descriptions. Refer to **Table 11-26** for TxFIFO status descriptions
7:0	RDATA	Receive data register. See **11.11.10 Receive Data Register (RDATA)**.

Table 11-22 RxFIFO Status

CMD 7	CMD 6	CMD 5	Description
0	0	0	RxFIFO Invalid or Empty. 1. Read of accompanying data byte is not valid. 2. RxFIFO invalid (flush message command in progress).
0	0	1	RxFIFO Contains More Than One Byte. 1. RxFIFO has between 2-12 bytes of data at the DLCMD2 access and no completion code.
0	1	0	RxFIFO Contains a Completion Code. 1. Indicates the presence of a completion code in the RxFIFO (not at head of RxFIFO) at the DLCMD2 access. There is no other data also in the RsFIFO.
0	1	1	Receive Data Byte in Postion 13 and No Completion Code in RxFIFO. 1. Position 13 in the RxFIFO is filled and no completion code is present.
1	0	0	RxFIFO Contains Exactly One Byte. 1. RxFIFO contains exactly one data byte (not a completion code).

Table 11-22 RxFIFO Status (Continued)

CMD 7	CMD 6	CMD 5	Description
1	0	1	Completion Code at Head of RxFIFO, More Bytes Available 1. RxFIFO has a completion code at the head, additional bytes from an in-progress message and no other completion codes.
1	1	0	Completion Code at Head of RxFIFO, Another Complete Message Available. 1. RxFIFO has a completion code at the head and another completed message (completion code presetn) in the RxFIFO.
1	1	1	Completion Code on at Head of FIFO. 1. RxFIFO has a completion code at the head and it is the only byte in the RxFIFO.

Table 11-23 Data Link Idle Status

STAT 4	Description
0	Data Link is Busy 1. A high level on the bus has been detected for more than 8 µs, or an EOF symbol is bing timed by the receive logic. Once bit 4 is a zero because of message activity it will not become one until an EOF symbol has been received. Noise on an idle line could cause the idle bit to change state as the idle bit is unfiltered. The order of events after a transmission is: EOD —> comp. code pushed —> interupt —> EOF (idle).
1	Data Link is Idle. 1. This is the point that a transmitter may begin accessing the bus.

Table 11-24 Transmitter Shorted Status

STAT 3	Description
0	Data Link is Not Shorted to Ground
1	Data Link is Shorted to Ground[1] 1. When a hardware fault prevents the active level from being sensed after the transmitter has driven the bus for 60 µs the “data link shorted” bit is set. The only way to clear this error and try again is to reset the transmitter with the DLCMD2 Command byte or a master reset of the system reset pin.

NOTE:

1. If the problem is a momentary short on the data link, it is appropriate to try reloading and transmitting the message again. If the problem is a defective line receiver then a retry could corrupt another mes-sage. The CPU must be able to handle this condition and execute a graceful recovery routine. A short to 12V will be sensed as a “data link is busy” condition. If the short to high is longer than the minimum time for a BREAK signal a completion code indicating a BREAK will be pushed into the RxFIFO after the short goes away.

Table 11-25 TxFIFO Underrunning Status

STAT 2	Description
0	The DLCMD2 TxFIFO in Not Underrunning
1	The DLCMD2 TxFIFO is Underrunning (TxFIFO is Half Empty, No Last Byte Indicated) 1. Cleared automatically following a read of the status register. 2. Actual TxFIFO underrun event (if it occurs) will be indicated in the completion code.

Table 11-26 TxFIFO Status

STAT 1	STAT 0	Description
0	0	TxFIFO is Empty
0	1	TxFIFO Contains Some Data Bytes. 1. TxFIFO contains between one and nine data bytes.
1	0	TxFIFO Almost Full. 1. TxFIFO contains 10 bytes (one byte left empty).
1	1	TxFIFO Full. 1. TxFIFO contains a valid “last byte” or it contains 11 bytes and no “last byte” (see Block Mode section).

11.11.10 Receive Data Register (RDATA)

RDATA— Receive Data Register 0xYF F60D

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
Status Byte Register (STAT)								RDATA							
RESET:															
0	0	0	0	0	0	0	0								

Table 11-27 RDATA Bit Descriptions

Bit(s)	Name	Description
15:8	STAT	Status byte. See **11.11.9 Status Register (STAT)**.
7:0	RDATA	Receive data register. The receive data register is used to read data from the RxFIFO. Data is popped from the RxFIFO on every read of the RDATA register. See **Figure 11-19**.

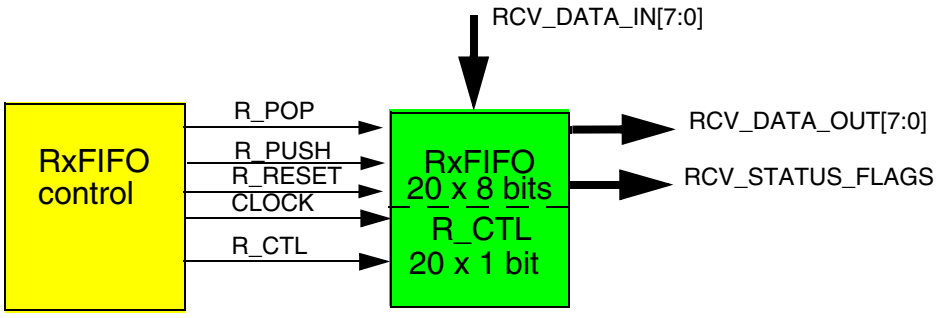


Figure 11-19 RxFIFO Block Diagram

11.11.11 Completion Code

When the message that is being received is complete, that is, after EOD has elapsed since the last transition on the bus, the DLCMD2 will prepare a completion code, which is a one byte descriptor with information about the message. If an error (CRC error, incomplete byte, bit timing error, underrun, and loss of arbitration) has occurred, the DLCMD2 will push a completion code into the RxFIFO after EOD has elapsed. If a BREAK symbol has been received, the DLCMD2 will push a completion code into the RxFIFO as soon as the bus is passive. Refer to **Figure 11-20**. This byte will be queued and transferred to the host in the same manner that a data byte is. The status byte will flag its presence.

NOTE

Any activity on the bus (i.e., noise pulses greater than 8 µs, messages, truncated messages, bad messages, etc.) will cause a completion code to be pushed into the RxFIFO after the bus is idle for an EOD length of time. In the case of noise, the completion code will not be pushed until the noise train is over for at least an EOD period of time and there will be only one completion code for all of the noise.

	7	6	5	4	3	2	1	0
	Errors in Receive Message	RxFIFO Overrun	Transmitter Actions		IFR	IFR w/CRC	Error Code	
RESET	0	0	0	0	0	0	0	0

Figure 11-20 Completion Code Byte Bit Definitions

Refer to **Table 11-28** for receive error descriptions. Refer to **Table 11-29** for RxFIFO overrun descriptions. Refer to **Table 11-30** for transmitter action descriptions. Refer to

Table 11-25 for TxFIFO underrunning status descriptions. Refer to Table 11-30 for IFR bit descriptions. Refer to Table 11-31 for IFR with/without a CRC bit descriptions. Refer to Table 11-32 for error code descriptions.

Table 11-28 Receive Error Status

Bit 7	Description
0	No Error Detected
1	Error(s) Detected 1. When set, certain errors occurred in the reception of this message (see description for bits 1 and 0). If not set, then bits 1 and 0 are also 0.

Table 11-29 RxFIFO Overrun Status

Bit 6	Description
0	No Overrun
1	Overrun 1. When set, a receiver RxFIFO overrun has occurred. It will be set as soon as the 19th data byte has been placed in the FIFO (20th position is always completion code). The host has not read out the DLCMD2 regularly enough to prevent the incoming message bytes from being lost. Previously received data bytes remain in the RxFIFO. The host must make a decision to disregard the partial message and perhaps transmit a request for retransmission of the message. If this bit is set no other bits in the completion code may be taken as valid.

Table 11-30 Transmitter Action Status

Bit 5	Bit 4	Description
0	0	Transmitter Not Involved 1. The transmitter did not attempt to contend against this message.
0	1	Transmitter Underrun 1. This message was not completed (not set if transmitter lost arbitration).
1	0	Transmitter Lost Arbitration 1.Transmitter contended against this message unsuccessfully or a RxFIFO overrun occurred (arbitration was not necessarily lost nor transmission stopped).
1	1	Transmitter Successful 1. Transmitter contended for this slot successfully (message transmitted successfully). Message originated from this node.

Table 11-31 IFR Bit Status

Bit 3	Description
0	Message not an IFR.
1	IFR 1. The preceding bytes that this completion code is associated with was an in-frame response.

Table 11-32 IFR With/Without a CRC Bit Status

Bit 2	Description
0	IFR Without CRC. 1. This in-frame response does not contain a CRC.
1	IFR With CRC. 1. The preceding byte(s) that this completion code is associated with was an in-frame response with a CRC.

Table 11-33 Error Code Status[1, 2, 3]

Bit 1	Bit 0	Description
0	0	CRC Error. 1. A CRC error was detected in the reception of this message or a receiver overrun occurred.
0	1	Incomplete Byte Received. 1. An incorrect mod 8 count was detected in the reception of this message. A complete byte was not received.
1	0	Bit Timing Error. 1. A bit timing error ocurred in this message. Will occur with a mismatch of internal clock divide relative to other nodes. The receiver immediately places this completion code after the reception of this error. May happen if a break occurs after the bus has been low for only 8 to 34 µs. In this case there will be two completion codes, the first for the bit error, and the second for the break. See Table 11-28 through Table 11-32 for othr bit timing errors.
1	1	Break Symbol Received. 1. A break was detected on the bus, or the bus was shorted high for more than 239 µs and has recovered. a. If the receive buffer has 19 bytes in it and a break is received, the completion code will indicate on overflow but no break. b. If the receive buffer has 20 bytes (full) and a break is received, thre will be no new completion code pushed and no indication of the break.

NOTE:

1. These error codes are used in conjunction with the error flag to report detected errors. Errors are reported based on the order of precedence. Only the highest precedence error be reported.
2. The following list is in precedence order, highest first:
   1. Bread condition occurred.
   2. Bit timing error detected.
   3. Incomplete byte (incorrect mod 8 count).
   4. CRC error detected.
   5. If any of these error conditions is noted, any pending EOD response message (in-frame response) will not be sent.

11.11.12 Bus Errors

The following figures describe various bit timing/ invalid symbol errors.

NOTE

TXP signals shown are what is expected if the error had not occurred and not necessarily what is on the bus during these error conditions.

Figure 11-21 shows an active symbol received during SOF transmission that is greater than the 8þs digital filter and less than the minimum SOF symbol time. This symbol will be flagged as an invalid symbol and the corresponding completion code will be $82.

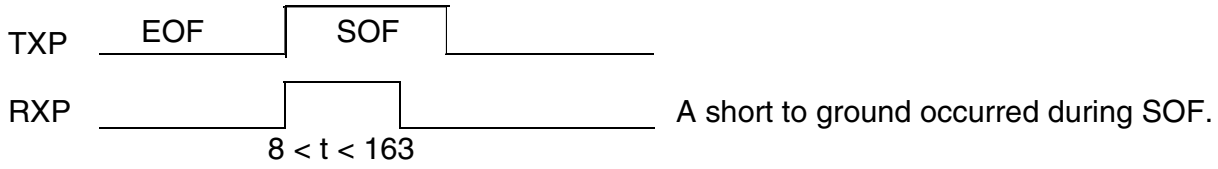


Figure 11-21 SOF Symbol Too Short

Figure 11-22 shows bit timing errors on short bits. This can be caused by noise, a bus short to ground (or VDD for passive bits), clock mismatches, etc. The corresponding completion code will be $82. If for the second case, the bus remains high for more than 239þs and then goes low, a completion code of $83 will follow.

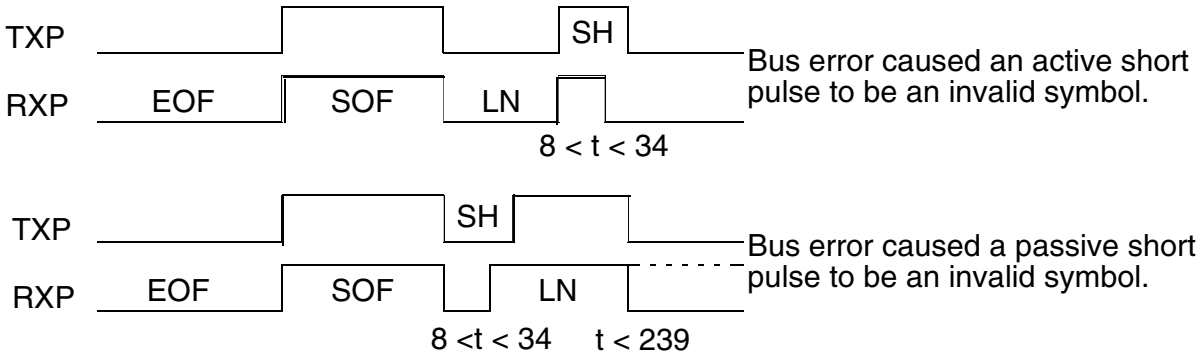
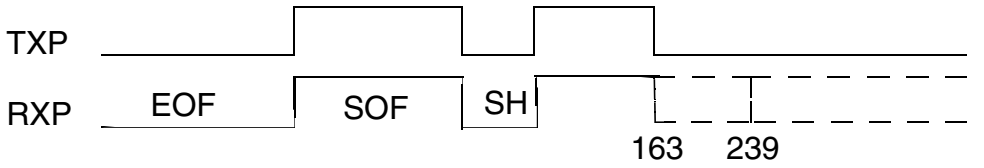


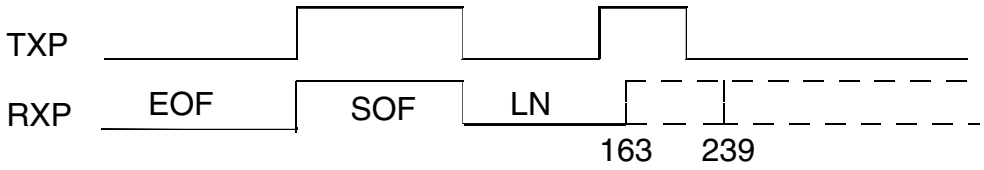
Figure 11-22 Figure 1-23 Short Bit Too Short

Figure 11-23 shows bit timing errors on long bits. This can be caused by noise, a bus short to V_{DD} (or ground for passive bits), clock mismatches, etc. For an active bit, if the bus goes low between 163 and 239 µs, the corresponding completion code will be $82.

If the bus goes low after the minimum BREAK time, a completion code of $83 will follow. For a passive bit, if the bus goes high between163 and 239 µs, the completion code will be $82. If the bus remains high for more than 239 µs and then goes low, a completion code of $83 will follow.



A short to V_{DD}, SOF (163 < t < 239) or BRK (t > 239) occurred in the middle of a message. All are errors.



A short to ground, EOD, or EOF occurred in the middle of a message.

Figure 11-23 Long Bit Too Long

Figure 11-24 shows bit timing errors on an EOD and EOF. This can be caused by noise, a bus short to V_{DD}, clock mismatches, etc. In these cases, the completion code will be $82.

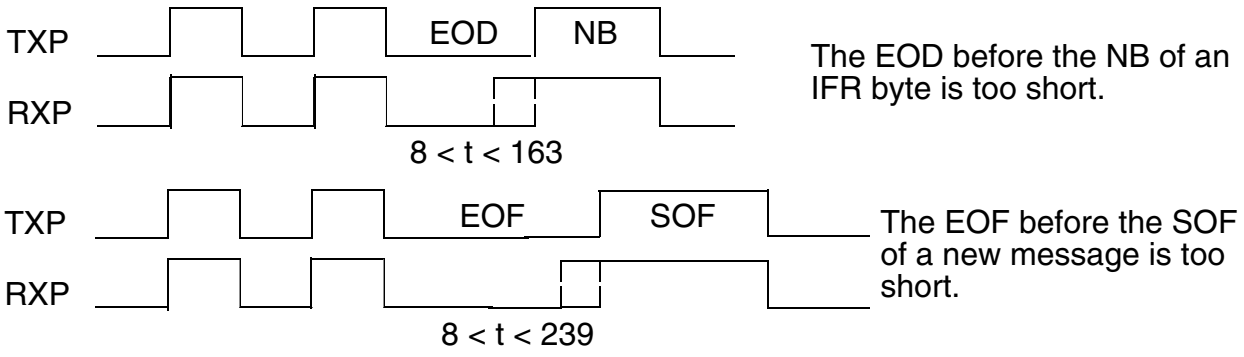


Figure 11-24 EOD and EOF Too Short

Figure 11-25 shows a bit timing error on a normalization bit. This can be caused by noise, a bus short to ground, clock mismatches, etc. The corresponding completion code will be $82.

Figure 11-25 Normalization Bit Too Short

Figure 11-26 shows a bit timing error on a normalization bit. This can be caused by noise, a bus short to V_{DD}, clock mismatches, etc. The corresponding completion code will be $82. If the bus remains high for more than 239 µs and then goes low, a completion code of $83 will follow.

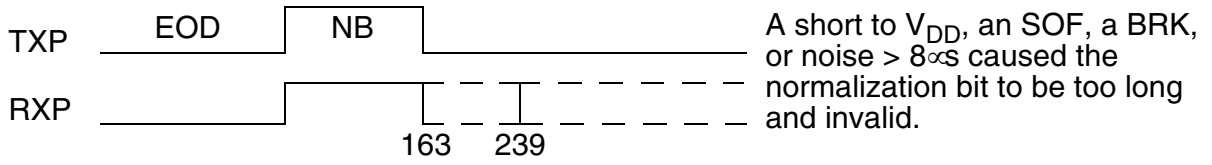


Figure 11-26 Normalization Bit Too Long

Other errors include CRC, incomplete byte, and BREAK. A CRC error occurs when the received message is corrupted by noise, delays, clock mismatches, etc. An incomplete byte error occurs when a received message ends on a non-byte boundary. This error also occurs when two extra 1's are transmitted after arbitration is lost on the last bit of a byte. Reception of a BREAK symbol is considered an error. This error occurs when the bus is held high for more than 239 µs and then goes low.

11.12 Mask Programmable Bus Error (BERR) Functionality

This DLCMD2 supports the BERR behavior according to IMB3 specification.

11.12.1 BERR_PLUG = 0

The DLCMD2 never asserts BERR signal in the IMB3.

11.12.2 BERR_PLUG = 1

The DLCMD2 will terminate the bus cycle with BERR in the following cases:

- Access to reserved 16-bit register within the DLCMD2's memory map.
- User access to supervisor-only registers when other registers in the DLCMD2's memory map are user-accessible (MCR SUPV bit is not set).
- Access to test register (TCR) when not in test mode.
- Writes to read-only registers.

11.13 Interrupt

This section describes DLCMD2 interrupt operation.

11.13.1 DLCMD2 Interrupts

In the default mode, interrupts may be requested due to one or more of four conditions becoming true. An optionally configured mode adds one more condition which will generate an interrupt. Interrupts may also be disabled.

If conditions which generate an interrupt occur while the DLCMD2 is being accessed, they will not be requested until the access is complete. Assuming interrupts are enabled, the default set of conditions that will cause the DLCMD2 module to request an interrupt are:

1. Completion code is placed in RxFIFO, DLCMD2 not accessed.
   a. An EOD has been received on the J1850 bus.
   b. bAdditional byte is received when the RxFIFO is full. This condition will cause the completion code to be pushed (in position 20 of the RxFIFO). The received byte and the rest of the message are ignored.
   c. DLCMD2 receives break. The reception of the break symbol will cause the completion code to be pushed into RxFIFO. If the break is received in the middle of DLCMD2 transmitting, it will stop transmitting, reset the transmitter, and clear the TxFIFO.
   d. DLCMD2 detects a bit-timing error. The bit-timing error will cause the completion code to be pushed into RxFIFO.
2. The RxFIFO has 12 bytes in it and the 13th byte is received, DLCMD2 not accessed. Completion code may or may not be present in RxFIFO. This differs from the 13th byte status indication which occurs only if there is no completion code in the RxFIFO.
   a. The status byte will only reflect this interrupt if there are no completion codes in the RxFIFO
   b. This interrupt generally means that the DLCMD2 is being neglected by the host.
3. A transmit operation is in progress, and there is no last byte to the message in the buffer and the buffer becomes half empty (six bytes left to transmit, five bytes in TxFIFO and the sixth byte is popped off to the transmit shift register), DLCMD2 not accessed.
4. The DLCMD2 is waking up on the rising edge of data link activity when it was previously in sleep mode.
   a. Any J1850 bus edge will wake up the DLCMD2. Will get a bit error indication if the DLCMD2 does not see at least 34 µs of the SOF.

By setting bit 15 in the ILR register in **11.11.3 Interrupt Level Register (ILR)**, one more condition capable of generating an interrupt is added:

5. A byte has been received into an empty RxFIFO, DLCMD2 not accessed.

Table 11-34 shows DLCMD2 interrupt operations. Refer to **Table 11-10** for IVR[2:0] Encoding.

Table 11-34 Interrupt Operations

Interrupt	Conditions Necessary to Enable/Re-Enable Interrupts	Conditions for Interrupt Assertion	Conditions to Clear this Interrupt
Wake-up	Enter low power mode	DLCMD2 module is in sleep mode. Positive going edge on bus > V_{ih} is sensed on bus.	Read IPR with bit 3 set and write IPR[3] “0” to clear
Transmitter half full (6 bytes)	1. “Load a byte into TxFIFO” with a fifth byte position filled, — OR — 2. Load in data bytes so that the sixth position in TxFIFO is occupied.	A transmit operation is in progress, no last byte indicated to the message in TxFIFO (*block mode*), TxFIFO becomes half empty, and transmit interrupt is enabled (*last byte not pushed into TxFIFO*)	Read IPR with bit 3 set and write IPR[3] “0” to clear
13th byte received	The only way to insure that this interruptis re-enabled is to complete empty out the Rx FIFO: 1. If 13th byte is occupeid and 14th isn’t a “flush byte” command will re-enable. — OR — 2. If 13th byte is occouped and 14th isn’t, and there is no completion code at the head of the RsFIFO, a “flush message” command will re-enable. — OR — 3. 13th byte becomes unoccupied.	RxFIFO receives 13th byte, (*Completion may or may not be present in RxFIFO, refer to* **11.13.1**)	Read IPR with bit 2 set and write IPR[2] “0” to clear.
EOD sensed on bus	Always enabled.	A completion code is placed onto the RxFIFO.	Read IPR with bit 1 set and write IPR[1] “0” to clear.
First byte received in an empty RxFIFO	1. If the first positon is filled, and the second position is empty, and a “flush byte” command is issued. — OR — 2. The first position becomes empty.	RxFIFO empty, first byte received, INTMODE in ILR must be set, and flush interrupt enabled (flush message except completion code, completion code will cauase EOD interrupt)	Read IPR with bit 0 set and write IPR[0] “0” to clear.

The DLCMD2 module is capable of generating one interrupt level on the IMB3. This level is programmed into the priority level bits in the interrupt level register (ILR[2:0]). This value determines which interrupt signal (IRQB7-1) is driven onto the bus during an interrupt request. IRQ0 is ignored by the CPU and so a zero priority level essentially disables all interrupts.

When an interrupt is requested, the CPU initiates an IACK cycle. The module decodes the IACK cycle and compares the CPU recognized level to the level that the module is currently requesting. If a match occurs, then arbitration begins. During arbitration, the arbitration number (refer to IARB[3:0] — Interrupt Arbitration ID (Bit 3 ~ Bit 0) in **11.11.1 Module Configuration Register (MCR)**) is driven in bit serial form, alternating between the IARB0 and IARB1 lines. The most significant bit of the arbitration number is driven first. Since the bus is a wire-AND one, a “low” level wins any contentions. Thus, the arbitration number is verified on a bit-by-bit basis. If contention is detected, (driving high and detecting low), then the module has lost the arbitration and immediately stop driving its arbitration number.

If the module wins the arbitration, it drives the eight bit interrupt vector stored in IVR register onto the data bus. The lower 3 bits are read-only bits and won’t be updated by the DLCMD2 until write-once to IVR occurs. These three bits represent the source of the interrupt as described in **Table 11-34**. The upper 5 bits are user programmable.

This interrupt generation scheme is shown in **Figure 11-27**.

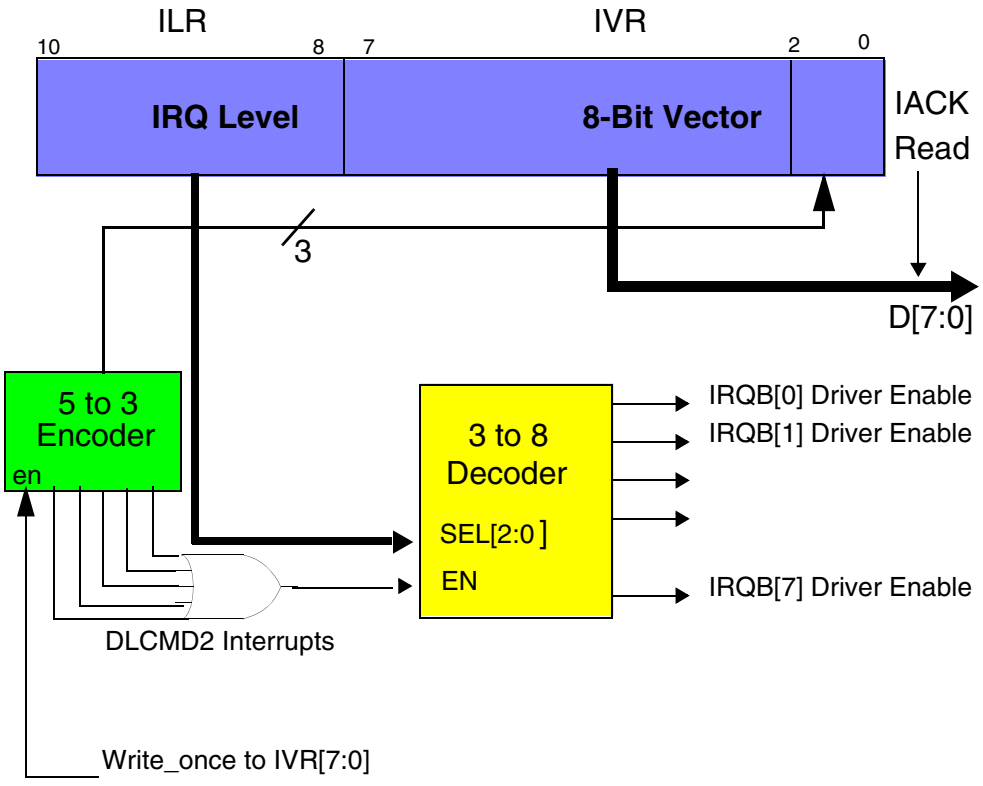


Figure 11-27 DLCMD2 Interrupt Vector Generation (IRQ_PLUG = 0)

11.14 In-Frame Response

This section describes how the DLCMD2 uses an in-frame response (IFR) in message transmission.

11.14.1 IFR Operation

The IFR may be sent by loading up a message for transmission with a “send on EOD” bit combination set in the command byte accompanying the first byte. The setting of the EOD bit combination will automatically next byte as part of, or the complete, response.

A message on the bus is determined to be ready for a response when an “end of data” (EOD) waveform has occurred on the bus. If a reply message initiates immediately following the EOD, then this will be an IFR. If no response is sent when EOD occurs, then an EOF will appear on the bus, signifying normal end of message without response. The EOF waveform is an extension of the EOD waveform. The reply will start with an active “1” or “0”, called the normalization bit, and may also have a CRC transmitted with it. The DLCMD2 is capable of arbitrating against and receiving all types of IFRs. Whether or not the received IFR contains a CRC or not is signalled through the use of the NBFS bit in the SCTL, see **11.11.4 Symbol Timing Control and Pre-Scaler Register (SCTL)**. IFRs are arbitrated just like any other message if more than one node wants to send a response. Single byte IFR from multiple responders (type 2) will retransmit if arbitration is lost. If arbitration is lost on the last bit of a type 2 IFR byte, two extra “1” bits will not be sent onto the bus since the IFR will retransmit. Upon beginning transmission of a single byte IFR from a single responder (type 1) or multiple byte IFR from a single responder (type 3), the DLCMD2 will enter TAR mode and not retransmit the IFR if arbitration is lost. If arbitration is lost on the last bit of a type 3 IFR byte, two extra “1” bits will be sent onto the bus.

This response message is sent only if:

1. There were no errors of any type in the original message requiring response. The only exception to this is that if an IFR is loaded after the transmitter lost arbitration to another message, the IFR will be sent at EOD.
2. IFR was loaded with a command byte that indicated either: “first byte” or “first and last” byte of a message.
3. This IFR was loaded into the DLCMD2 after the start of the message was detected, and t_{resp} before the EOD was recognized.

When the DLCMD2 receives the command byte and data byte signaling an IFR, it will start transmitting the response after EOD even if there is no last byte in the TxFIFO. The DLCMD2 will start the response by automatically transmitting an active phase “0” or a “1” depending on how the NBFS bit is programmed.

It is left to the host to remember that any messages that were in the transmit buffer at the time the IFR was loaded have been flushed, and must be reloaded by the host for transmission. This is true even if the IFR was not successfully loaded or transmitted.

Completion code bit 3 will inform the host of whether or not an IFR has occurred. If an IFR has occurred, the RxFIFO will contain the initial message, with a completion code, and the response, with its completion code. Each completion code identifies the message associated with it as a normal message, or a response. The only way that the CPU can tell that such a sequence has occurred on the bus is by reading the completion code.

If any errors occur during the reception of an IFR:

1. A completion code indicating an error occurred during reception is pushed into the RxFIFO.
2. Anything left of the IFR is not received off the bus.

If bit 3 of the completion code is set, then bit 2 will indicate whether the response had a CRC field or not. If bit 2 is set, then the response had a CRC field. This is important for the CPU to determine the meaning of the response.

A DLCMD2 may send an IFR to an IFR with CRC but no response may follow an IFR without CRC.

For the system to enter a mode that allows an IFR, some coordination of nodes is required. The node sending an IFR must select the appropriate interrupt configuration (interrupt on first byte), and be ready to transfer a response into the DLCMD2, signalling through the command byte that this is an IFR.

11.14.2 IFR Abort Conditions

Table 11-35 shows conditions in which IFRs are aborted.he IFR was l

Table 11-35 IFR Aborted Conditions

IFR Transmission Aborted If	Actions Taken
Host underran transmitter (no “last byte” indicated in the TxFIFO at EOD, and host did not supply a last byte in time for transmission).	DLCMD2 inverts the CRC field on the truncated IFR.
IFR loses contention to another IFR.	DLCMD2 stops transmitting multiple byte IFRs, and flushes the TxFIFO immediately. DLCMD2 can be configured to auto-retry or stop transmission of single byte IFRs upon loss of arbitration.
Error detected in the received message.	DLCMD2 resets the transmitter and TxFIFO, no IFR is sent.
Error in IFR during transmission.	IFR is stopped, transmitter and TxFIFO are reset.
IFR not loaded correctly. 1. Bus idle when IFR was loaded (loaded before any message is on the bus). 2. IFR loaded less than t_{resp} before EOD is sensed on bus (loaded too late). 3. IFR not loaded with “first byte” indicated. 4. IFR bit set in command byte, but no data is loaded at all.	DLCMD2 resets the transmitter and TxFIFO, no IFR is sent.

11.14.3 IFR Types

The DLCMD2 supports types 1, 2 and 3 IFRs with arbitration and auto-retransmission. See **Figure 11-28** for a description of each type of IFR. The following sections describe how the DLCMD2 can be configured to transmit the three types of IFRs.

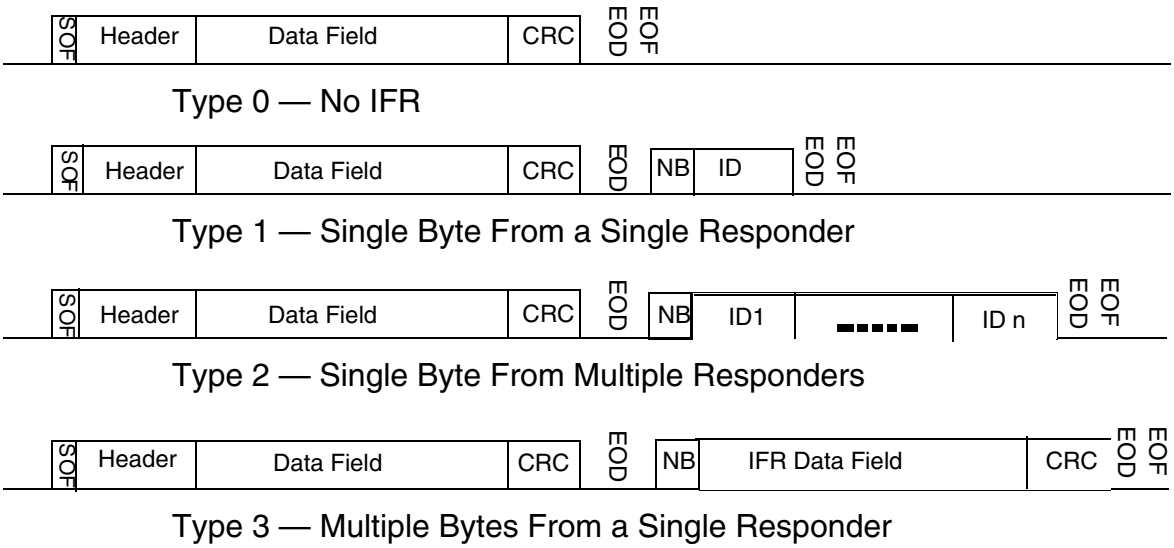


Figure 11-28 Types of IFR

11.14.3.1 Type 1 IFR

A type 1 IFR is a single byte IFR without CRC transmitted by a single responder. If arbitration is lost, the DLCMD2 will not attempt to re-transmit the IFR byte in the TxFIFO. The TxFIFO will be cleared and the IFR byte will be lost.

The DLCMD2 can be configured to transmit a type 1 IFR by writing an IFR byte into the TDATA register and a command byte of $BC indicating “send on EOD without CRC” and “load as first and last byte.”

11.14.3.2 1.Type 2 IFR

A type 2 IFR results from multiple responders transmitting a single byte IFR onto the bus. Since only one IFR will win arbitration, the other responders will re-transmit their IFRs as soon as the IFR winning arbitration finishes. The IFR byte in the TxFIFO will not be cleared until transmission is successful or an error is detected on the bus.

The DLCMD2 can be configured to transmit a type 2 IFR by writing an IFR byte into the TDATA register and a command byte of $DC indicating “send on EOD with auto re-try” and “load as first and last byte.”

11.14.3.3 Type 3 IFR

A type 3 IFR is a multiple byte IFR with or without CRC transmitted by a single responder. If arbitration is lost, the DLCMD2 will not attempt to re-transmit the IFR bytes in the TxFIFO. The TxFIFO will be cleared and the IFR bytes will be lost.

The DLCMD2 can be configured to transmit a type 3 IFR with CRC by writing the first IFR byte into the TDATA register and a command byte of $74 indicating “send on EOD with CRC” and “load as first byte.”

The DLCMD2 can also be configured to transmit a type 3 IFR without CRC by writing the first byte into the TDATA register and a command byte of $B4 indicating “send on EOD without CRC” and “load as first byte.”troller Module (DLCMD2)

11.15 System Overview

Typical usage of the DLCMD2 in a microcomputer system is shown in **Figure 11-29**.

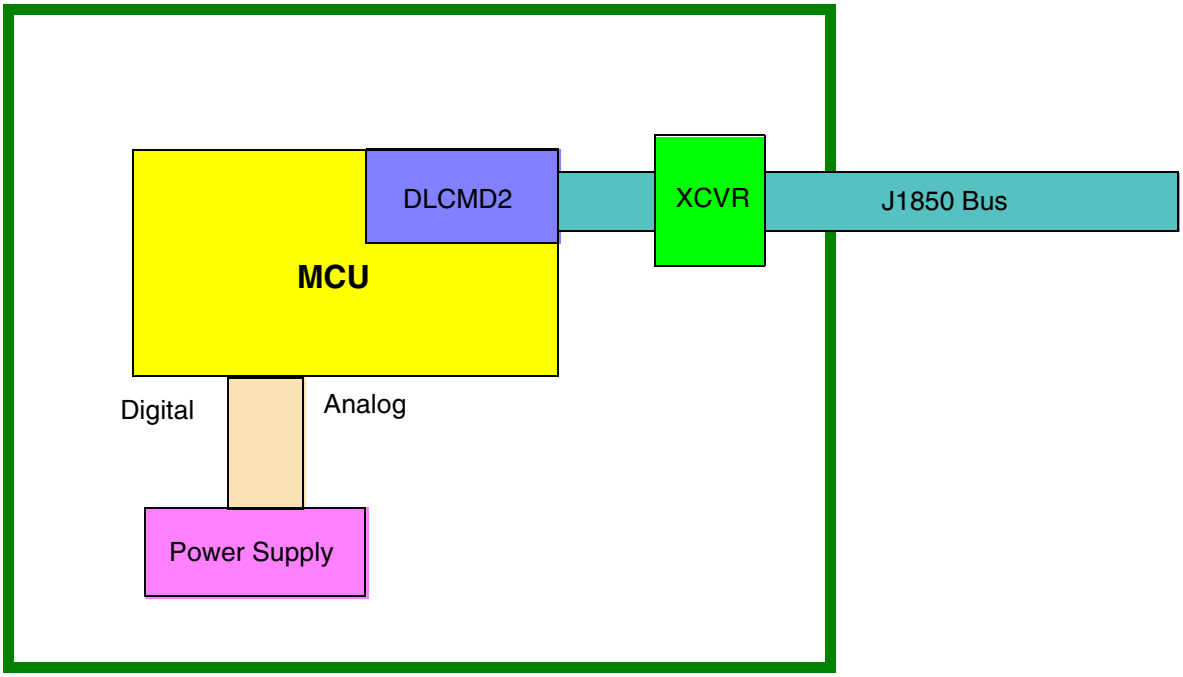


Figure 11-29 J1850 Node

11.16 Data Link Controller Module (DLCMD2) — Internal

11.16.1 Inter-Module Bus 3 (IMB3)

11.16.2 Bus Signals

The DLCMD2 shall have an internal interface compatible with the IMB3. The address and data bus along with their associated control and handshake lines are used to transfer data between the IMB3 and the DLCMD2. **Table 11-36** lists the IMB3 signals used in the DLCMD2.

Table 11-36 IMB3 Signals

Signal Name	Description	Input / Output	Assert / Negate
ICLOCK	System Clock	I	—
IMSTRSTB	Master Reset	I	CLOCK HIGH / CLOCK LOW
ISYSRSTB	System Reset	I	CLOCK HIGH / CLOCK LOW
IFREEZEB	Freeze	I	CLOCK LOW
ITSTMODB	Test Mode	I	CLOCK LOW
IMODMAP	Module Memory Mapping	I	B4
IFC[2:0]	Function Codes	I	B4 / B3 + B3A
ICYSB	Cycle Start	I	B4 / B3 + B3A
IWRITEB	Read/$\overline{\text{Write}}$	I	B4 / B3 + B3A
IADDR[23:0]	Address Bus	I	B4 / B3 + B3A
IASB	Address Strobe	I	B1 / B4
IAACKB	Address Knowledge	O	B1 / B4
ISIZ[1:0]	Size Codes	I	B4 / B3 + B3A
IDSB	Data Strobe	I	B2 / B4 * B1
IILBS[1:0]	interrupt Level Byte Select	I	CLOCK HIGH
IIMB3TSTB	Test Enable	I	CLOCK HIGH
ICLKDIS	Clock Disable	I	CLOCK LOW
ICLKE	Supports IMB 2X Bus Clock 2x bus clock is not used in this module. This signal is terminated in the BIU with a std clk rcd.	I	—
ICLK2XE	Supports IMB 2X Bus Clock 2x bus clock is not used in this module. This signal is terminated in the BIU with a std clk rcd.	I	—
IDATA[15:0]	Data Bus	I/O	B2(Write) B3(Read / Write B4(Read) / B4* + B1 + B3 following the assertion of BTACK
IDTACKB	Data Transfer Acknowledge	O	B2 + B3 / B4* + B1
IBERRB	Bus Error	I/O	B2 + B3 / B4* + B1
IIRQB[7:0]	Interupt Request Level	O	CLOCK LOW / CLOCK HIGH
IIARB[1:0]	Interrupt Arbitration	I/O	Bit0:CLOCK LOW / CLOCK HIGH Bit1:CLOCK HIGH / CLOCK LOW

11.17 Module I/O Signals

11.17.1 Signal Descriptions

This section provides information on external signal functions.

11.17.2 External Connections

Figure 11-30 shows DLCMD2 external connections.

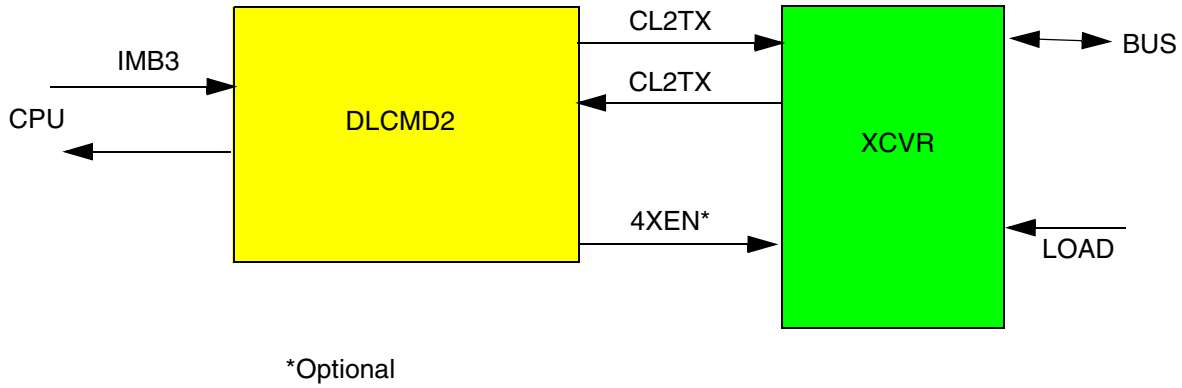


Figure 11-30 DLCMD2 External Connections

11.17.3 Signal Functions

Table 11-37 summarizes DLCMD2 external signals.

Table 11-37 Signal Names

Module Signal Name (External)	Input / Output	Description
CL2TX	O	DLCMD2 digital output to transceiver
CL2RX	I	DLCMD2 digital input from transceiver
4XEN[1]	O	4X transmit enable output to transceiver

NOTE:
1. May be provided by the DLCMD2, CPU, or other IC discrete. The DLCMD2 module will provide this signal to the periphery of the module. This way, an on-chip transceiver may be more easilty accommodated.

11.17.3.1 CL2TX

This pin is a logic level output. A logic “0” output drives the BUS output to 0 VDC (external pull down resistor to ground) and a logic “1” output produces a high voltage at the bus output. An internal 200 KΩ to ground in the XCVR guarantees a logic “0” input when this pin is open circuit (the bus output is tri-stated).

11.17.3.2 CL2RX

This is a CMOS compatible input used to get receiver data to the microprocessor. When the voltage on BUS is over 3.5 ± .2 VDC, this pin shall be a logic “1.” When the voltage on BUS is under 3.5 ± .2 VDC, this pin shall be a logic “0.” There is a minimum of.1 VDC of hysteresis between the bus high and low (and vice versa) transition points.

11.17.3.3 4XEN

This is an optional pin used to select whether waveshaping for the J1850 output is enabled or disabled. A logic “0” shall enable waveshaping and a logic “1” shall disable

waveshaping. An internal 200 KΩ to ground in the XCVR guarantees a logic “0” input when this pin is open circuit.

SECTION 12
STATIC RANDOM ACCESS MEMORY (SRAM)

12.1 Introduction

This SRAM module is a fast access (two clocks) general purpose 8 Kbytes (8,192 bytes) static RAM (SRAM) for the MCU and is accessed via the IMB3. In addition there is 2 Kbytes (configured as four blocks of 512 bytes each) of patch static RAM. These modules are fast access (two clocks) general purpose static RAM (SRAM) for the MCU with a patch option which provides a method to overlay the internal CMFI memory for emulation. As an additional feature, the 512-byte arrays can be used as additional SRAM. A register map showing the SRAM and overlay configuration registers and memory blocks is shown in **Figure 12-1**.

The SRAM module is powered by V_{DDL} in normal operation and may be used as standby SRAM if standby power is supplied via the V_{STBY} pin of the MCU. Switching between V_{DDL} and V_{STBY} will occur automatically.

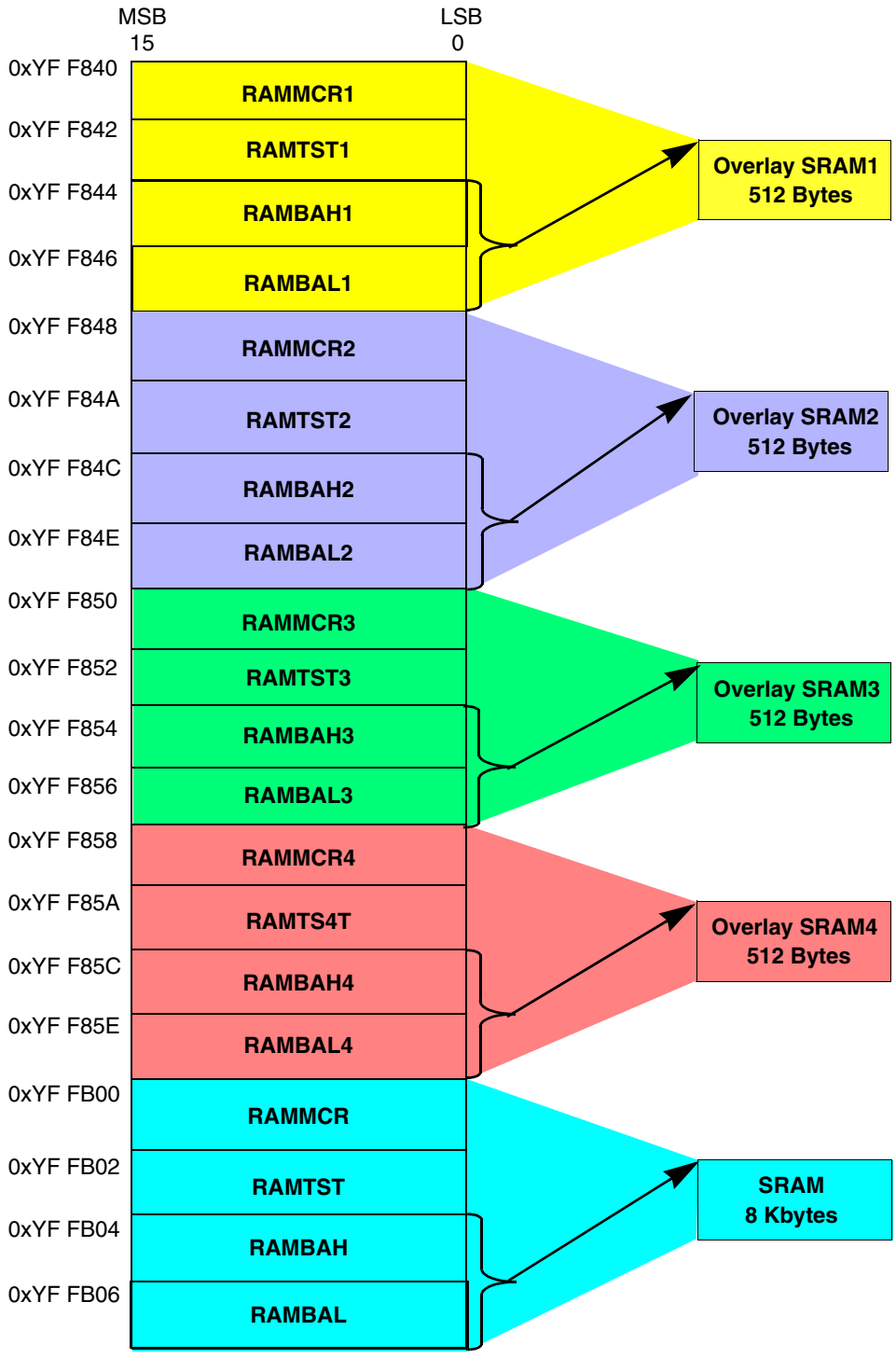
When used as general purpose SRAM, this module is accessed via the IMB3. The SRAM may be read or written as either bytes or words. Access for aligned long-word operations is supported by back-to-back IMB3 accesses (four clocks) to accommodate 32-bit operations.

12.2 Programmer's Model

Each SRAM module consists of two separately addressable sections. The first is a set of memory mapped control and status registers used for configuration and testing of the SRAM array. The second section is the array itself.

12.2.1 SRAM Control Block

There are four registers provided for configuration and control of each SRAM module: SRAM module configuration register (RAMMCR), a factory test register (RAMTST), and the array base address registers (RAMBAH, RAMBAL). In order to offer the maximum protection for the SRAM array, the SRAM module control registers are located in supervisor data space.



Y = M111, where M is the logic state of the module mapping (MM) bit in the SCIM2E configuration register (SCIMMCR).

Figure 12-1 SRAM Module Configuration

12.2.2 SRAM Array

The SRAM array itself can be placed anywhere in the address map of the MCU by means of the array base address registers. The high order address lines (IADDR[31:N] N=13 and nine for 8-Kbyte and 512-byte arrays) are compared with the value in RAMBAH and RAMBAL registers for a base address match. This value points to the lowest address that SRAM data may be located and is always on an 8-Kbyte or 512-byte boundary. The SRAM array top is on a 256-byte boundary. The only restrictions on the base address is that it must be on an address boundary greater than or equal to the array size. If the SRAM array base address is located to overlap the SRAM control block then access to the 8 bytes in the SRAM array located at the same address as the SRAM control block are ignored allowing the control block to be accessed.

NOTE

This is only the eight bytes in the SRAM control block; this mapping may have unknown results for other modules.

12.2.2.1 SRAM Array Addressing

The BIU of the SRAM module compares IADDR[31:N] (N = 13 for 8 Kbytes and 9 for 512-byte arrays) of the IMB3 with the value of the array base address registers. If they match then the low address and ISIZ[1:0] are used to access the SRAM location in the array. Addresses in the array that are not implemented will be ignored by the SRAM module allowing an external device to respond to the address. Function codes are also checked for the correct access rights. If the array is placed in supervisor space, user accesses will be ignored allowing an external device to respond to the address. If the array is placed in unrestricted space, it will respond to both user and supervisor accesses. The array may also be placed in program space, or program/data space. Program space allows the SRAM array to contain only program instructions for execution, or program counter relative addressing modes for operand fetches from the array. Program/data space allows both program and data may be stored in the SRAM array.

12.3 SRAM Module Control and Status Registers

The SRAM module control and status registers are used to control and monitor the operation of the SRAM array. The following sections describe the operation of each register in the SRAM control block. Since all the registers are in supervisor data space, the only way to change the state of the registers is through a supervisor program. Any other restrictions for making changes to the registers will be noted in the description of the register. Unless otherwise noted, any source of master reset will cause register bits to be forced to their reset state; while, system resets have no effect on the registers. Reads to unimplemented bits will return “0”; while, writes have no effect.

12.3.1 Module Configuration Register (RAMMCR)

All modules on the MCU contain a module configuration register. The RAMMCR register bits configure the SRAM module for stop operation and for proper access rights to the array.

RAMMCR1 — SRAM Module Configuration Register **0xYF F840**
RAMMCR2 **0xYF F848**
RAMMCR3 **0xYF F850**
RAMMCR4 **0xYF F858**
RAMMCR **0xYF FB00**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
STOP	RESERVED		PDS	RLCK	0	RASP[1:0]		RESERVED							
RESET:															
1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

Table 12-1 RAMMCR Bit Settings

Bit(s)	Name	Description
15	STOP	Stop control. The assertion of the STOP control bit in the RAMMCR register by a bus master signals the SRAM module to enter into the STOP state. When STOP is asserted, SRAM array accesses are ignored. When the SRAM module is in normal mode of operation the array base address registers are write protected. 0 = SRAM module normal operation. 1 = Causes SRAM module to enter low power stop mode.
14:13	—	Reserved
12	PDS	Power down status. PDS is a optional status bit in the RAMMCR that enables a power monitor for the SRAM array. The power monitor circuit will clear the PDS bit (PDS = “0”) if the array standby power is lost. If the PDS bit is unimplemented reads will return “0”. 0 = Power monitor for the SRAM array is disabled. SRAM array standby power has failed. 1 = Power monitor for the SRAM array is enabled. SRAM array standby power has not failed.
11	RLCK	Base address lock. 0 = SRAM base address registers are writable from the IMB3. 1 = SRAM base address registers are write locked.
10	—	Reserved
9:8	RASP[1:0]	Array space. The RASP field limits access to the SRAM array to one of four CPU32 address spaces. Refer to. 0 = Only the module configuration register, test register, and interrupt register are designated as supervisor-only data space. Access to all other locations is unrestricted. 1 = All QADC64 registers and tables are designated as supervisor-only data space.
7:0	—	Reserved

Table 12-2 RASP Encoding

RASP[1:0]	Space
00	Unrestricted program and data
01	Unrestricted program
10	Supervisor program and data
11	Supervisor program

12.3.2 Array Base Address Registers (RAMBAH, RAMBAL)

The array base address registers are provided to allow the flexibility of placing the SRAM array anywhere in the memory map. RAMBAH and RAMBAL contains an address field used to specify the most significant bits of the lowest address value in

the SRAM array address block. The SRAM array base address is placed on a 512, 1-Kbyte, 2-Kbyte or 4-Kbyte block boundary (block size greater than or equal to the size of the array). RAMBAH and RAMBAL may only be written while the SRAM is in STOP mode STOP = “1” and the lock bit is not set RLCK = “0”. Once the RLCK bit is set, writes will have no effect on RAMBAH and RAMBAL. This will prevent runaway software from inadvertently re-mapping the array. The SRAM must be in STOP mode while RAMBAH or RAMBAL are written this prevents inadvertent intermediate array mapping to be acknowledged.

RAMBAH1, RAMBAL1 — SRAM Base Address Registers **0xYF F844, 0xYF F846**
RAMBAH2, RAMBAL2 0xYF F84C, 0xYF F84E
RAMBAH3, RAMBAL3 0xYF F854, 0xYF F856
RAMBAH4, RAMBAL4 0xYF F85C, 0xYF F85E
RAMBAH, RAM1BAL 0xYF FB04, 0xYF FB06

MSB 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	LSB 16
RAMBAH															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RAMBAL							RESERVED								
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-3 RAMBAH, RAMBAL Bit Settings

Bit(s)	Name	Description
31:16	RAMBAH	Array base address high. With STOP asserted the base Address field of RAMBAH may be changed so that the array may be placed at the desired address in the memory map. This must be done by a supervisor program since the register is in supervisor data space. To lock the base address field, RLCK in the RAMMCR should be set. This will prevent the base address field from being changed until the next master reset.
15:9	RAMBAL	Array base address low. With STOP asserted the base Address field of RAMBAL may be changed so that the array may be placed at the desired address in the memory map. This must be done by a supervisor program since the register is in supervisor data space. To lock the base address field, RLCK RAMMCR should be set. This will prevent the base address field from being changed until the next master reset.
8:0	—	Reserved

12.4 Operation

The SRAM module has several modes of operation. The following sections describe SRAM module operation in each of these modes.

12.4.1 Normal Operation

Normal operation is when the SRAM may be accessed via the IMB3 by a bus master and is being powered by V_{DDL}. The array may be accessed as byte or word. Access may be either read or write.

12.4.1.1 Read/Write

The SRAM module allows a byte or aligned word read/write in one IMB3 bus cycle. Long word read/write will require an additional bus cycle. An IMB3 bus cycle requires two system clocks.

Table 12-4 SRAM Array Read/Write Minimum Access Times

TYPE	Bus Cycles Required for Read or Write	Number of System Clocks
Byte	1	2
Aligned Word	1	2
Aligned Long Word	2	4

12.4.2 Standby Operation

A separate supply pin is used by the standby SRAM module to maintain the contents of the SRAM array during a power down phase. The external supply pin of the MCU is known as V_{STBY}. Data in the standby SRAM will be retained down to the lowest supply voltage, either V_{DDL} or V_{STBY}, see **APPENDIX E ELECTRICAL AND AC CHARACTERISTICS**. Circuitry within the standby SRAM module will automatically switch between V_{DDL} and V_{STBY}. The SRAM module will switch to standby power when $V_{DDL} < V_{STBY} - V_{SWITCH}$.

When the SRAM array is powered by the V_{STBY} pin of the MCU, access to the SRAM array is blocked. Data read from the SRAM array during this condition will not be valid. Data written to the SRAM may be corrupted if switching occurs during a write operation. For the module to function correctly as general purpose SRAM, the maximum value for $V_{STBY} \leq V_{DDL}$.

12.4.2.1 Power Down

In order to guarantee valid standby SRAM data during power down, external low voltage inhibit circuitry, (external to the MCU), must be designed to force the RESET pin into the active state before V_{DDL} drops below its normal limit. This is necessary to inhibit a write cycle to the SRAM during power down.

12.4.3 RESET Operation

When a synchronous reset occurs, a bus master will be allowed, as a result of internal synchronization, to complete the current access. Thus, a write bus cycle, byte or word, that is in progress when a synchronous reset occurs will be completed without error. During the RESET state, once an in-progress write has been completed, further writes to the standby SRAM array will be inhibited.

Note that a long word write will be completed coherently only if the reset occurs during the second write bus cycle. If reset occurs during the first write bus cycle, only the first word will be written to the SRAM array and the second write will not be allowed to occur. In this case, the long word data contained in the SRAM will not be coherent. The first word will contain the most significant half of the new long word information and the second word will contain the least significant half of the old long word information.

If a reset is generated by an asynchronous reset such as the loss of clocks or software watchdog time-out, the contents of the standby SRAM array are not guaranteed.

12.4.4 STOP Operation

The assertion of the STOP control bit of the RAMMCR (see **12.3.1 Module Configuration Register (RAMMCR)**) causes the SRAM module to enter its lowest power consuming state. The register block may still be accessed to allow the STOP control bit to be cleared and the array base address registers to be updated, see **12.3.2 Array Base Address Registers (RAMBAH, RAMBAL)**.

When in stop mode, the SRAM array can not be read or written. All data in the array will be retained. Switching to V_{STBY} will occur as normal if V_{DDL} drops below its specified value when the SRAM module is in stop mode.

12.4.5 Overlay Operation

The four 512-byte SRAM blocks can be used independently of the main 8K SRAM array. The blocks can be initialized to be continuous with the main array or can be used to overlay the flash module. The overlay feature is enabled whenever an overlay SRAM module base address is mapped over the flash array address space. The 512-byte SRAM block should be placed on a 512-byte boundary and will respond to any access to the overlayed section of the flash and will disable the flash contents from being read.

APPENDIX A
INTERNAL MEMORY MAP

The tables below use the following notations.

In the Access column:

S = Supervisor Access Only
U = User Access
T = Test Access

In the Reset column:

A = Affected by $\overline{\text{RESET}}$
U = Unchanged
X = Unknown

The codes in the Reset column indicate which reset has an effect on register values.

INDEX of MEMORY MAP TABLES

Table A-1 QADC64 (1) (Queued Analog-to-Digital Converter)

Address	Access	Symbol	Register	Size	Reset
0xYF F000	S	QADC64MCR	QADC64 Module Configuration Register. See **Table 8-7** for bit descriptions.	16	X
0xYF F002	T	QADC64TEST	QADC64 Test Register	16	X
0xYF F004	S	QADC64INT	Interrupt Register. See **Table 8-8** for bit descriptions.	16	X
0xYF F006	S	PORTQA/ PORTQB	Port A and Port B Data. See **Table 8-9** for bit descriptions.	16	X
0xYF F008	S	DDRQA	Port A Data Direction Register. See **Table 8-10** for bit descriptions.	16	X
0xYF F00A	S	QACR0	QADC64 Control Register 0. See **Table 8-11** for bit descriptions.	16	X
0xYF F00C	S	QACR1	QADC64 Control Register 1. See **Table 8-12** for bit descriptions.	16	X
0xYF F00E	S	QACR2	QADC64 Control Register 2. See **Table 8-14** for bit descriptions.	16	X
0xYF F010	S	QASR0	QADC64 Status Register 0. See **Table 8-16** for bit descriptions.	16	X
0xYF F012	S	QASR1	QADC64 Status Register 1. See **Table 8-18** for bit descriptions.	16	X
0xYF F014 – 0xYF F0FE	—	—	Reserved	—	—

Table A-2 TouCAN (1) (CAN 2.0B Controller)

0xYF F100 — 0xYF F10F	S/U	MBUFF0	TouCAN Message Buffer 0. See **Table 10-5** and **Table 10-6** for message buffer definitions.	—	U
0xYF F110 — 0xYF F11F	S/U	MBUFF1	TouCAN Message Buffer 1. See **Table 10-4** and **Table 10-2** for message buffer definitions.	—	U
0xYF F120 — 0xYF F12F	S/U	MBUFF2	TouCAN Message Buffer 2. See **Table 10-4** and **Table 10-2** for message buffer definitions.	—	U
0xYF F130 — 0xYF F13F	S/U	MBUFF3	TouCAN Message Buffer 3. See **Table 10-4** and **Table 10-2** for message buffer definitions.	—	U
0xYF F140 — 0xYF F14F	S/U	MBUFF4	TouCAN Message Buffer 4. See **Table 10-4** and **Table 10-2** for message buffer definitions.	—	U
0xYF F150 — 0xYF F15F	S/U	MBUFF5	TouCAN Message Buffer 5. See **Table 10-4** and **Table 10-2** for message buffer definitions.	—	U
0xYF F160 — 0xYF F16F	S/U	MBUFF6	TouCAN Message Buffer 6. See **Table 10-4** and **Table 10-2** for message buffer definitions.	—	U
0xYF F170 — 0xYF F17F	S/U	MBUFF7	TouCAN Message Buffer 7. See **Table 10-4** and **Table 10-2** for message buffer definitions.	—	U
0xYF F180 — 0xYF F18F	S/U	MBUFF8	TouCAN Message Buffer 8. See **Table 10-4** and **Table 10-2** for message buffer definitions.	—	U

Table A-2 TouCAN (1) (CAN 2.0B Controller) (Continued)

0xYF F190 — 0xYF F19F	S/U	MBUFF9	TouCAN Message Buffer 9. See **Table 10-4** and **Table 10-2** for message buffer definitions.	—	U
0xYF F1A0 — 0xYF F1AF	S/U	MBUFF10	TouCAN Message Buffer 10. See **Table 10-4** and **Table 10-2** for message buffer definitions.	—	U
0xYF F1B0 — 0xYF F1BF	S/U	MBUFF11	TouCAN Message Buffer 11. See **Table 10-4** and **Table 10-2** for message buffer definitions.	—	U
0xYF F1C0 — 0xYF F1CF	S/U	MBUFF12	TouCAN Message Buffer 12. See **Table 10-5** and **Table 10-6** for message buffer definitions.	—	U
0xYF F1D0 — 0xYF F1DF	S/U	MBUFF13	TouCAN Message Buffer 13. See **Table 10-4** and **Table 10-2** for message buffer definitions.	—	U
0xYF F1E0 — 0xYF F1EF	S/U	MBUFF14	TouCAN Message Buffer 14. See **Table 10-4** and **Table 10-2** for message buffer definitions.	—	U
0xYF F1F0 — 0xYF F1FF	S/U	MBUFF15	TouCAN Message Buffer 15. See **Table 10-4** and **Table 10-2** for message buffer definitions.	—	U

Table A-3 QADC64 (2) (Queued Analog-to-Digital Converter)

0xYF F300 – 0xYF F37F	S	CCW	QADC64 Conversion Command Word Table. See **Table 8-19** for bit descriptions.	16	X
0xYF F380 – 0xYF F3FE	S	RJURR	QADC64 Result Word Table Right-Justified, Unsigned Result Register. See **8.12.10 Conversion Command Word Table** for bit descriptions.	16	X
0xYF F300 – 0xYF F37E	S	LJSRR	QADC64 Result Word Table Left-Justified, Signed Result Register. See **8.12.11 Result Word Table** for bit descriptions.	16	X
0xYF F380 – 0xYF F3FE	S	LJURR	QADC64 Result Word Table Left-Justified, Unsigned Result Register. See **8.12.11 Result Word Table** for bit descriptions.	16	X

Table A-4 QSM_B (Queued Serial Module B)

Address	Access	Symbol	Register	Size	Reset
0xYF F400	S	QSMCR_B	QSM Module Configuration Register. See **Table 7-3** for bit descriptions.	16	X
0xYF F402	T	QTEST_B	QSM Test Register.	16	X
0xYF F404	S	QILR/QIVR_B	QSM Interrupt Level/interrupt Vector Register. See **Table 7-8** and **Table 7-9** for bit descriptions.	16	X
0xYF F406	S	—	Reserved	—	X
0xYF F408	S	SCCR0_B	SCI1 Control Register 0. See **Table 7-22** for bit descriptions.	16	X
0xYF F40A	S	SCCR1_B	SCI1Control Register 1. See **Table 7-24** for bit descriptions.	16	X

Table A-4 QSM_B (Queued Serial Module B) (Continued)

Address	Access	Symbol	Register	Size	Reset
0xYF F40C	S	SCSR_B	SCI1 Status Register. See Table 7-26 for bit descriptions.	16	X
0xYF F40E	S	SCDR_B	SCI1 Data Register. See Table 7-27 for bit descriptions.	16	X
0xYF F410 – 0xYF F412	S	—	Reserved	—	X
0xYF F414	S	PORTQS_B	QSMCM Port QS Data Register. See 7.2.3.1 QSM Port Data Register (PORTQS) for bit descriptions.	16	X
0xYF F416	S	PQSPAR_B / DDRQST_B	QSMCM Port QS Pin Assignment Register/ QSMCM Port QS Data Direction Register. See Table 7-11 and Table 7-13 for bit descriptions.	16	X
0xYF F418	S	SPCR0_B	QSPI Control Register 0. See Table 7-14 for bit descriptions.	16	X
0xYF F41A	S	SPCR1_B	QSPI Control Register 1. See Table 7-16 for bit descriptions.	16	X
0xYF F41C	S	SPCR2_B	QSPI Control Register 2. See Table 7-17 for bit descriptions.	16	X
0xYF F41E	S	SPCR3_B / SPSR_B	QSPI Control Register 3/QSPI Status Register. See Table 7-18 and Table 7-19 for bit descriptions.	16	X
0xYF F420 — 0xYF F4FF	S	—	Reserved	16	X
0xYF F500 — 0xYF F51F	S	SCRQ_B	QSPI Receive Queue Locations. See 7.4.4 Receiver Operation for bit descriptions.	16	X
0xYF F520 — 0xYF F53F	S	SCTQ_B	QSPI Transmit Queue Locations. See 7.4.3 Transmitter Operation for bit descriptions.	16	X
0xYF F540 — 0xYF F54F	S	QSPIRAM_B	QSPI Transmit Queue Locations. See Table 7-20 for bit descriptions.	16	X

Table A-5 DLCMD (Data Link Controller Module)

Address	Access	Symbol	Register	Size	Reset
0xYF F600	S	MCR	DLCMD Module Configuration Register See Table 11-6 for bit descriptions.	16	X
0xYF F602	S	—	Reserved	16	X
0xYF F604	S	IPR	DLCMD Interrupt Pending Register See Table 11-8 for bit descriptions.	16	X
0xYF F606 / 0xYF F607	S	ILR / IVR	DLCMD Interrupt Level Register / Interrupt Vector Register See Table 11-9 and Table 11-10 for bit descriptions.	16	X
0xYF F608	S / U	SCTL	DLCMD Symbol Timing Control and Pre-Scaler Register See Table 11-11 for bit descriptions.	16	X
0xYF F60A	S / U	SDATA	DLCMD Symbol Timing Data Register See Table 11-13 for bit descriptions.	16	X

Table A-5 DLCMD (Data Link Controller Module) (Continued)

Address	Access	Symbol	Register	Size	Reset
0xYF F60C	S / U	CMD/TDATA	DLCMD Command Register / Transmit Data Register See **Table 11-15** and **Table 11-18** for bit descriptions.	16	X
0xYF F60D/ 0xYF F60E	S	RDATA/STAT	DLCMD Receive Data Register / Status Register / See **Table 11-27** and **Table 11-21** for bit descriptions.	16	X

Table A-6 DPTRAM (Dual-Port TPU RAM)

Address	Access	Symbol	Register	Size	Reset
0xYF F680	S	DPTMCR	DPT module configuration register. See **Table 6-2** for bit descriptions.	16	X
0xYF F682	T	DPTTCR	DPT test register.	16	X
0xYF F684	S	DPTBAR	DPT array address register. See **Table 6-3** for bit descriptions.	16	X
0xYF F686	S	MISRH	DPT multiple input signature register high. See **6.4.4 MISR High (MISRH) and MISR Low (MISRL)** for bit descriptions.	16	X
0xYF F688	S	MISRL	DPT multiple input signature register low. See **6.4.4 MISR High (MISRH) and MISR Low (MISRL)** for bit descriptions.	16	X
0xYF F68A	S	MISCNT	DPT multiple input signature counter. See **6.4.5 MISC Counter (MISCNT)** for bit descriptions.	16	X

Table A-7 FASRAM Module (Fast Access Static Random Access Memory)

Address	Access	Symbol	Register	Size	Reset
0xYF F6C0	S	FMCR	FASRAM module configuration register. See **Table 4-2** for bit descriptions.	16	X
0xYF F6C2	S	FTEST	FASRAM module test register. See **Table 4-1** for bit descriptions.	16	X
0xYF F6C4	S	FBAR-H	FASRAM base address register. See **Table 4-3** for bit descriptions.	16	X
0xYF F6C6	S	FBAR-L	FASRAM base address register. See **Table 4-3** for bit descriptions.	16	X
0xYF F6C8	S	FCCR0	FASRAM comparator value 0 register. See **Table 4-4** for bit descriptions.	16	X
0xYF F6CA	S	FCCR1	FASRAM comparator value 1 Register. See **Table 4-4** for bit descriptions.	16	X
0xYF F6CE	S	FMATCH	FASRAM most recent match address Register. See **4.9.3 Most Recent Match Address (FMATCH)** for bit descriptions.	16	X
0xYF F610	S	FSTATUS	FASRAM most recent match status register. See **Table 4-1** for bit descriptions.	16	X
0xYF F612	S	—	Reserved.	16	X
0xYF F61E	S	—	Reserved.	16	X

Table A-8 CTM9 (Configurable Timer Module)

Address	Access	Symbol	Register	Size	Reset
BIUSM (CTM9 Bus Interface Unit Submodule)					
0xYF F700	S	BIUMCR	BIUSM Module Configuration Register. See **Table 9-19** and **9.8.4.1 BIUMCR — BIUSM Module Configuration Register** for bit descriptions.	16	X
0xYF F702	T	BIUTEST	BIUSM Test Register.	16	X
0xYF F704	S	BIUTBR	BIUSM Time Base Register. See **9.8.4.2 BIUTBR — BIUSM Time Base Register** for bit descriptions.	16	X
CPSM (CTM9 Counter Prescaler Submodule)					
0xYF F708	S	CPCR	CPSM Control Register. See **Table 9-21** and **9.9.2.1 CPCR — CPSM Control Register** for bit descriptions.	16	X
0xYF F70A	T	CPTR	CPSM Test Register.	16	X
MCSM2 (CTM9 Modulus Counter Submodule 2)					
0xYF F710	S	MCSM2SIC	MCSM2 Status/Interrupt/Control Register. See **Table 9-5** for bit descriptions.	16	X
0xYF F712	S	MCSM2CNT	MCSM2 Counter Register. See **9.3.2 The MCSM Counter** for bit descriptions.	16	X
0xYF F714	S	MCSM2ML	MCSM2 Modulus Latch Register. See **9.3.11 MCSMML — MCSM Modulus Latch Register** for bit descriptions.	16	X
DASM3 (CTM9 Double-Action Submodule 3)					
0xYF F718	S	DASM3SIC	DASM3 Status/Interrupt/Control Register. See **Table 9-11** and **9.5.5.1 DASMSIC — DASM Status/Interrupt Control Register** for bit descriptions.	16	X
0xYF F71A	S	DASM3A	DASM3 Register A. See **9.5.5.2 DASMA — DASM Data Register A** for bit descriptions.	16	X
0xYF F71C	S	DASM3B	DASM3 Register B. See **9.5.5.3 DASMB — DASM Data Register B** for bit descriptions.	16	X
DASM4 (CTM9 Double-Action Submodule 4)					
0xYF F720	S	DASM4SIC	DASM4 Status/Interrupt/Control Register. See **Table 9-11** and **9.5.5.1 DASMSIC — DASM Status/Interrupt Control Register** for bit descriptions.	16	X
0xYF F722	S	DASM4A	DASM4 Register A. See **9.5.5.2 DASMA — DASM Data Register A** for bit descriptions.	16	X
0xYF F724	S	DASM4B	DASM4 Register B. See **9.5.5.3 DASMB — DASM Data Register B** for bit descriptions.	16	X
PWM5 (CTM9 Pulse Width Modulation Submodule 5)					
0xYF F728	S	PWM5SIC	PWM5 Status, Interrupt and Control Register. See **Table 9-15** and **9.7.13.1 PWMSIC — Status, Interrupt and Control Register** for bit descriptions.	16	X

Table A-8 CTM9 (Configurable Timer Module) (Continued)

Address	Access	Symbol	Register	Size	Reset
0xYF F72A	S	PWM5A	PWM5 Period Register. See **9.7.13.2 PWMA — PWM Period Register** for bit descriptions.	16	X
0xYF F72C	S	PWM5B	PWM5 Pulse Width Register. See **9.7.13.3 PWMB — PWM Pulse Width Register** for bit descriptions.	16	X
0xYF F72E	S	PWM5C	PWM5 Counter Register. See **9.7.13.4 PWMC — PWM Counter Register** for bit descriptions.	16	X
PWM6 (CTM9 Pulse Width Modulation Submodule 6)					
0xYF F730	S	PWM6SIC	PWM6 Status, Interrupt and Control Register. See **Table 9-15** and **9.7.13.1 PWMSIC — Status, Interrupt and Control Register** for bit descriptions.	16	X
0xYF F732	S	PWM6A	PWM6 Period Register. See **9.7.13.2 PWMA — PWM Period Register** for bit descriptions.	16	X
0xYF F734	S	PWM6B	PWM6 Pulse Width Register. See **9.7.13.3 PWMB — PWM Pulse Width Register** for bit descriptions.	16	X
0xYF F736	S	PWM6C	PWM6 Counter Register. See **9.7.13.4 PWMC — PWM Counter Register** for bit descriptions.	16	X
PWM7 (CTM9 Pulse Width Modulation Submodule 7)					
0xYF F738	S	PWM7SIC	PWM7 Status, Interrupt and Control Register. See **Table 9-15** and **9.7.13.1 PWMSIC — Status, Interrupt and Control Register** for bit descriptions.	16	X
0xYF F73A	S	PWM7A	PWM7 Period Register. See **9.7.13.2 PWMA — PWM Period Register** for bit descriptions.	16	X
0xYF F73C	S	PWM7B	PWM7 Pulse Width Register. See **9.7.13.3 PWMB — PWM Pulse Width Register** for bit descriptions.	16	X
0xYF F73E	S	PWM7C	PWM7 Counter Register. See **9.7.13.4 PWMC — PWM Counter Register** for bit descriptions.	16	X
PWM8 (CTM9 Pulse Width Modulation Submodule 8)					
0xYF F740	S	PWM8SIC	PWM8 Status, Interrupt and Control Register. See **Table 9-15** and **9.7.13.1 PWMSIC — Status, Interrupt and Control Register** for bit descriptions.	16	X
0xYF F742	S	PWM8A	PWM8 Period Register. See **9.7.13.2 PWMA — PWM Period Register** for bit descriptions.	16	X
0xYF F744	S	PWM8B	PWM8 Pulse Width Register. See **9.7.13.3 PWMB — PWM Pulse Width Register** for bit descriptions.	16	X
0xYF F746	S	PWM8C	PWM8 Counter Register. See **9.7.13.4 PWMC — PWM Counter Register** for bit descriptions.	16	X

Table A-8 CTM9 (Configurable Timer Module) (Continued)

Address	Access	Symbol	Register	Size	Reset
DASM9 (CTM9 Double-Action Submodule 9)					
0xYF F748	S	DASM9SIC	DASM9 Status/Interrupt/Control Register. See **Table 9-11** and **9.5.5.1 DASMSIC — DASM Status/Interrupt Control Register** for bit descriptions.	16	X
0xYF F74A	S	DASM9A	DASM9 Register A. See **9.5.5.2 DASMA — DASM Data Register A** for bit descriptions.	16	X
0xYF F74C	S	DASM9B	DASM9 Register B. See **9.5.5.3 DASMB — DASM Data Register B** for bit descriptions.	16	X
DASM10 (CTM9 Double-Action Submodule 10)					
0xYF F750	S	DASM10SIC	DASM10 Status/Interrupt/Control Register. See **Table 9-11** and **9.5.5.1 DASMSIC — DASM Status/Interrupt Control Register** for bit descriptions.	16	X
0xYF F752	S	DASM10A	DASM10 Register A. See **9.5.5.2 DASMA — DASM Data Register A** for bit descriptions.	16	X
0xYF F754	S	DASM10B	DASM10 Register B. See **9.5.5.3 DASMB — DASM Data Register B** for bit descriptions.	16	X
MCSM11 (CTM9 Modulus Counter Submodule 11)					
0xYF F758	S	MCSM11SIC	MCSM11 Status/Interrupt/Control Register. See **Table 9-5** for bit descriptions.	16	X
0xYF F75A	S	MCSM11CNT	MCSM11 Counter Register. See **9.3.2 The MCSM Counter** for bit descriptions.	16	X
0xYF F75C	S	MCSM11ML	MCSM11 Modulus Latch Register. See **9.3.11 MCSMML — MCSM Modulus Latch Register** for bit descriptions.	16	X
FCSM (CTM9 Free Running Counter Submodule)					
0xYF F760	S	FCSMSIC	FCSM Status, Interrupt and Control Register. See **Table 9-2** for bit descriptions.	16	X
0xYF F762	S	FCSMCNT	FCSM Counter Register. See **9.2.7.2 FCSMCNT — FCSM Counter Register** for bit descriptions.	16	X
SASM14 (CTM9 Single-Action Submodule 14)					
0xYF F770	S	S14ICA	SASM14 Status/Interrupt/Control Register A. See **Table 9-7** and **9.4.4.1 SICA — SASM Status/Interrupt Control Register A** for bit descriptions.	16	X
0xYF F772	S	S14DATA	SASM14 Data Register A. See **9.4.4.2 SDATA — SASM Data Register A** for bit descriptions.	16	X
0xYF F774	S	S14ICB	SASM14 Status/Interrupt/Control Register B. See **9.4.4.3 SICB — SASM Status/Interrupt Control Register B** for bit descriptions.	16	X
0xYF F776	S	S14DATB	SASM14 Data Register B. See **9.4.4.4 SDATB — SASM Data Register B** for bit descriptions.	16	X

Table A-8 CTM9 (Configurable Timer Module) (Continued)

Address	Access	Symbol	Register	Size	Reset
SASM16 (CTM9 Single-Action Submodule 16)					
0xYF F780	S	S16ICA	SASM16 Status/Interrupt/Control Register. See **Table 9-7** and **9.4.4.1 SICA — SASM Status/Interrupt Control Register A** for bit descriptions.	16	X
0xYF F782	S	S16DATA	SASM16 Data Register. See **9.4.4.2 SDATA — SASM Data Register A** for bit descriptions.	16	X
0xYF F784	S	S16ICB	SASM16 Status/Interrupt/Control Register B. See **9.4.4.3 SICB — SASM Status/Interrupt Control Register B** for bit descriptions.	16	X
0xYF F786	S	S16DATB	SASM16 Data Register B. See **9.4.4.4 SDATB — SASM Data Register B** for bit descriptions.	16	X
SASM18 (CTM9 Single-Action Submodule 18)					
0xYF F790	S	S18ICA	SASM18 Status/Interrupt/Control Register. See **Table 9-7** and **9.4.4.1 SICA — SASM Status/Interrupt Control Register A** for bit descriptions.	16	X
0xYF F792	S	S18DATA	SASM18 Data Register. See **9.4.4.2 SDATA — SASM Data Register A** for bit descriptions.	16	X
0xYF F794	S	S18ICB	SASM18 Status/Interrupt/Control Register B. See **9.4.4.3 SICB — SASM Status/Interrupt Control Register B** for bit descriptions.	16	X
0xYF F796	S	S18DATB	SASM18 Data Register B. See **9.4.4.4 SDATB — SASM Data Register B** for bit descriptions.	16	X
SASM20 (CTM9 Single-Action Submodule 20)					
0xYF F7A0	S	S20ICA	SASM20 Status/Interrupt/Control Register. See **Table 9-7** and **9.4.4.1 SICA — SASM Status/Interrupt Control Register A** for bit descriptions.	16	X
0xYF F7A2	S	S20DATA	SASM20 Data Register. See **9.4.4.2 SDATA — SASM Data Register A** for bit descriptions.	16	X
0xYF F7A4	S	S20ICB	SASM20 Status/Interrupt/Control Register B. See **9.4.4.3 SICB — SASM Status/Interrupt Control Register B** for bit descriptions.	16	X
0xYF F7A6	S	S20DATB	SASM20 Data Register B. See **9.4.4.4 SDATB — SASM Data Register B** for bit descriptions.	16	X

Table A-9 TPU3_B (1) (Time Processor Unit)

Address	Access	Symbol	Register	Size	Reset
0xYF F800	S	TPUMCR_B	TPU3 module configuration register See **Table 5-7** for bit descriptions.	16	X
0xYF F802	T	TCR_B	TPU3 test configuration register	16	X
0xYF F804	S	DSCR_B	Development support control register See **Table 5-8** for bit descriptions.	16	X

Table A-9 TPU3_B (1) (Time Processor Unit) (Continued)

Address	Access	Symbol	Register	Size	Reset
0xYF F806	S	DSSR_B	Development support status register See Table 5-9 for bit descriptions.	16	X
0xYF F808	S	TICR_B	TPU3 interrupt configuration register See Table 5-10 for bit descriptions.	16	X
0xYF F80A	S	CIER_B	Channel interrupt enable register See Table 5-11 for bit descriptions.	16	X
0xYF F80C	S	CFSR0_B	Channel function selection register 0 See Table 5-12 for bit descriptions.	16	X
0xYF F80E	S	CFSR1_B	Channel function selection register 1 See Table 5-12 for bit descriptions.	16	X
0xYF F810	S	CFSR2_B	Channel function selection register 2 See Table 5-12 for bit descriptions.	16	X
0xYF F812	S	CFSR3_B	Channel function selection register 3 See Table 5-12 for bit descriptions.	16	X
0xYF F814	S	HSQR0_B	Host sequence register 0 See Table 5-13 for bit descriptions.	16	X
0xYF F816	S	HSQR1_B	Host sequence register 1 See Table 5-13 for bit descriptions.	16	X
0xYF F818	S	HSRR0_B	Host service request register 0 See Table 5-13 for bit descriptions.	16	X
0xYF F81A	S	HSRR1_B	Host service request register 1 See Table 5-14 for bit descriptions.	16	X
0xYF F81C	S	CPR0_B	Channel priority register 0 See Table 5-15 for bit descriptions.	16	X
0xYF F81E	S	CPR1_B	Channel priority register 1 See Table 5-15 for bit descriptions.	16	X
0xYF F820	S	CISR_B	Channel interrupt status register See Table 5-17 for bit descriptions.	16	X
0xYF F822	T	LR_B	Link register	16	X
0xYF F824	T	SGLR_B	Service grant latch register	16	X
0xYF F826	T	DCNR_B	Decoded channel number register	16	X
0xYF F828	S	TPUMCR2_B	TPU module configuration register 2 See Table 5-18 for bit descriptions.	16	X
0xYF F82A	S	TPUMCR3_B	TPU module configuration 3 See Table 5-21 for bit descriptions.	16	X
0xYF F82C	T	ISDR_B	Internal scan data register	16	X
0xYF F82E	T	ISCR_B	Internal scan control register	16	X

Table A-10 SRAM Modules (1)
(Static Random Access Memory)

Address	Access	Symbol	Register	Size	Reset
SRAM(A)					
0xYF F840	S	RAMMCR1	SRAM1 Module Configuration Register. See Table 12-1 for bit descriptions.	16	X
0xYF F842	T	RAMTST1	SRAM1 Test Register.	16	X
0xYF F844	S	RAMBAH1	SRAM1 Base Address High Register. See Table 12-3 for bit descriptions.	16	X
0xYF F846	S	RAMBAL1	SRAM1 Base Address Low Register. See Table 12-3 for bit descriptions.	16	X

Table A-10 SRAM Modules (1) (Static Random Access Memory) (Continued)

Address	Access	Symbol	Register	Size	Reset
SRAM(B)					
0xYF F848	S	RAMMCR2	SRAM2 Module Configuration Register. See **Table 12-1** for bit descriptions.	16	X
0xYF F84A	T	RAMTST2	SRAM2 Test Register.	16	X
0xYF F84C	S	RAMBAH2	SRAM2 Base Address High Register. See **Table 12-3** for bit descriptions.	16	X
0xYF F84E	S	RAMBAL2	SRAM2 Base Address Low Register. See **Table 12-3** for bit descriptions.	16	X
SRAM(C)					
0xYF F850	S	RAMMCR3	SRAM3 Module Configuration Register. See **Table 12-1** for bit descriptions.	16	X
0xYF F852	T	RAMTST3	SRAM3 Test Register.	16	X
0xYF F854	S	RAMBAH3	SRAM3 Base Address High Register. See **Table 12-3** for bit descriptions.	16	X
0xYF F856	S	RAMBAL3	SRAM3 Base Address Low Register. See **Table 12-3** for bit descriptions.	16	X
SRAM(D)					
0xYF F858	S	RAMMCR4	SRAM4 Module Configuration Register. See **Table 12-1** for bit descriptions.	16	X
0xYF F85A	T	RAMTST4	SRAM4 test register.	16	X
0xYF F85C	S	RAMBAH4	SRAM4 base address high register. See **Table 12-3** for bit descriptions.	16	X
0xYF F85E	S	RAMBAL4	SRAM4 base address low register. See **Table 12-3** for bit descriptions.	16	X

Table A-11 TPU3_B (2) (Time Processor Unit)

0xYF F900 – 0xYF F90F	S	—	Channel 0 parameter registers	16	X
0xYF F910 – 0xYF F91F	S	—	Channel 1 parameter registers	16	X
0xYF F920 – 0xYF F92F	S	—	Channel 2 parameter registers	16	X
0xYF F930 – 0xYF F93F	S	—	Channel 3 parameter registers	16	X
0xYF F940 – 0xYF F94F	S	—	Channel 4 parameter registers	16	X
0xYF F950 – 0xYF F95F	S	—	Channel 5 Parameter Registers	16	X
0xYF F960 – 0xYF F96F	S	—	Channel 6 parameter registers	16	X
0xYF F970 – 0xYF F97F	S	—	Channel 7 parameter registers	16	X
0xYF F980 – 0xYF F98F	S	—	Channel 8 parameter registers	16	X
0xYF F990 – 0xYF F99F	S	—	Channel 9 parameter registers	16	X
0xYF F9A0 – 0xYF F9AF	S	—	Channel 10 parameter registers	16	X

Table A-11 TPU3_B (2) (Time Processor Unit) (Continued)

0xYF F9B0 – 0xYF F9BF	S	—	Channel 11 parameter registers	16	X
0xYF F9C0 – 0xYF F9CF	S	—	Channel 12 parameter registers	16	X
0xYF F9D0 – 0xYF F9DF	S	—	Channel 13 parameter registers	16	X
0xYF F9E0 – 0xYF F9EF	S	—	Channel 14 parameter registers	16	X
0xYF F9F0 – 0xYF F9FF	S	—	Channel 15 parameter registers	16	X

Table A-12 BIM Module

Address	Access	Symbol	Register	Size	Reset
0xYF FA00	S	MCR	Module configuration register. See **Table 3-8** for bit descriptions.	16	X
0xYF FA02	S	MTR	BIM module test register. See **Table 3-8** for bit descriptions.	16	X
0xYF FA04	S	BIMTR/MDR	BIM test register/Module disable register See **Table 3-10** for bit descriptions.	16	X
0xYF FA06	S	—	Reserved.	16	X
0xYF FA08	S	SYNCR	Clock synthesizer control register. See **Table 3-62** for bit descriptions.	16	X
0xYF FA0A	S	SYNST/RSR	Clock synthesizer status/reset status registers. See **Table 3-63** and **Table 3-65** for bit descriptions.	16	X
0xYF FA0C	S	—	Reserved	16	X
0xYF FA0E	T	PCON	Port/clock configuration shadow register. See **Table 3-11** for bit descriptions.	16	X
0xYF FA10	S	PORTA/PORTB	Port A output data / port B output data register. See **Table 3-14** amd **Table 3-16** for bit descriptions.	16	X
0xYF FA12	S	PORTAP/ PORTBP	Port A pin data / port B pin data register. See **Table 3-14** and **Table 3-16** for bit descriptions.	16	X
0xYF FA14	S	DDRAB	Port A/B data direction register. See **Table 3-13** for bit descriptions.	16	X
0xYF FA16	T	—	Reserved	16	X
0xYF FA18	S	PORTC/PORTD	Port C output data / port D output data register. See **Table 3-22** and **Table 3-24** for bit descriptions.	16	X
0xYF FA1A	S	PORTCP/ PORTDP	Port C pin data register / port D pin data register. See **Table 3-23** and **Table 3-28** for bit descriptions.	16	X
0xYF FA1C	S	DDRC/DDRD	Port C data direction / port D data direction register. See **Table 3-21** and **Table 3-26** for bit descriptions.	16	X

Table A-12 BIM Module (Continued)

Address	Access	Symbol	Register	Size	Reset
0xYF FA1E	T	PCPAR/PDPAR	Port C pin assignment / port D pin assignment register. See Table 3-19 and Table 3-25 for bit descriptions.	16	X
0xYF FA20	S	PORTK/PORTE	Port K output data / port E output data register. See Table 3-53 and Table 3-32 for bit descriptions.	16	X
0xYF FA22	S	PORTKP/ PORTEP	Port K pin data / port E pin data register. See Table 3-54 and Table 3-33 for bit descriptions.	16	X
0xYF FA24	S	DDRK/DDRE	Port K direction / port E direction register. See Table 3-52 and Table 3-31 for bit descriptions.	16	X
0xYF FA26	T	PKPAR/PEPAR	Port K pin assignment / port E pin assignment register. See Table 3-51 and Table 3-30 for bit descriptions.	16	X
0xYF FA28	S	PORTG/PORTH	Port G output data / port H output data register. See Table 3-45 and Table 3-48 for bit descriptions.	16	X
0xYF FA2A	S	PORTGP/ PORTHP	Port G pin data / port H pin data register. See Table 3-46 and Table 3-49 for bit descriptions.	16	X
0xYF FA2C	S	DDRG/DDRH	Port G data direction / port H data direction register. See Table 3-44 and Table 3-47 for bit descriptions.	16	X
0xYF FA2E	S	—	Reserved	16	X
0xYF FA30	S	PORTF	Port F output data. See Table 3-38 for bit descriptions.	16	X
0xYF FA32	S	PORTFP	Port F pin data register. See Table 3-39 for bit descriptions.	16	X
0xYF FA34	S	DDRF	Port F data direction register. See Table 3-37 for bit descriptions.	16	X
0xYF FA36	S	PFPAR	Port F pin assignment register. See Table 3-36 for bit descriptions.	16	X
0xYF FA38	S	PORTFE	Port F edge flags register. See Table 3-40 for bit descriptions.	16	X
0xYF FA3A	S	PFIACK/PFEER	Port F IACK response / port F edge-detect enable register. See Table 3-41 for bit descriptions.	16	X
0xYF FA3C		PFLVR	Port F level register. See Table 3-42 for bit descriptions.		
0xYF FA3E		—	Reserved		
0xYF FA40		MSRA	Test module master shift A register. See Table 3-7 for bit descriptions.		
0xYF FA42		MSRB	Test module master shift B register. See Table 3-7 for bit descriptions.		
0xYF FA44		SCRA	Test module shift count A register. See Table 3-7 for bit descriptions.		
0xYF FA46		REPS	Test module repetition counter register. See Table 3-7 for bit descriptions.		

Table A-12 BIM Module (Continued)

Address	Access	Symbol	Register	Size	Reset
0xYF FA48		CREG	Test module control register register. See Table 3-7 for bit descriptions.		
0xYF FA4A		DREG	Test module distributed register. See Table 3-7 for bit descriptions.		
0xYF FA4C		BCSOR2	Burst chip select option register 2 register. See Table 3-7 for bit descriptions.		
0xYF FA4E		—	Reserved.		
0xYF FA50		SYPCR	System protect control register. See Table 3-67 for bit descriptions.		
0xYF FA52		TIC	Timer control register. See Table 3-68 for bit descriptions.		
0xYF FA53		TIV	Timer interrupt vector register. See Table 3-69 for bit descriptions.		
0xYF FA54		—	Reserved		
0xYF FA55		SWS	SWDOG service register. See Table 3-70 for bit descriptions.		
0xYF FA56		PRE	Prescaler (read-only) register. See Table 3-71 for bit descriptions.		
0xYF FA58		SWI	Software watchdog internal register See Table 3-72 for bit descriptions.		
0xYF FA5A		RTI	Real-time interval register. See Table 3-73 for bit descriptions.		
0xYF FA5C		SWIT	Software watchdog interval timer operation register. See Table 3-74 for bit descriptions.		
0xYF FA5E		RTDC	Real-time downcounter (read-only) register. See Table 3-7 for bit descriptions.		
0xYF FA5F		RTIT	Real-time interval counter and RTC operation register. See Table 3-75 for bit descriptions.		
0xYF FA60		CSBAR1	Chip select base address register 1. See Table 3-80 for bit descriptions.		
0xYF FA62		CSOR1	Asynchronous chip select option register 1. See Table 3-81 for bit descriptions.		
0xYF FA64		CSBAR2	Chip select base address register 2. See Table 3-80 for bit descriptions.		
0xYF FA66		CSOR2	Chip select option register 2. See Table 3-80 for bit descriptions.		
0xYF FA68		CSBAR3	Chip select base address register 3. See Table 3-80 for bit descriptions.		
0xYF FA6A		CSOR3	Chip select option register 3. See Table 3-80 for bit descriptions.		
0xYF FA6C		CSBAR4	Chip select base address register 4. See Table 3-80 for bit descriptions.		
0xYF FA6E		CSOR4	Chip select option register 4. See Table 3-80 for bit descriptions.		
0xYF FA70		CSBAR5	Chip select base address register 5. See Table 3-80 for bit descriptions.		
0xYF FA72		CSOR5	Chip select option register 5. See Table 3-80 for bit descriptions.		
0xYF FA74		CSBAR6	Chip select base address register 6. See Table 3-80 for bit descriptions.		

Table A-12 BIM Module (Continued)

Address	Access	Symbol	Register	Size	Reset
0xYF FA76		CSOR6	Chip select option register 6. See Table 3-80 for bit descriptions.		
0xYF FA78		CSBAR7	Chip select base address register 7. See Table 3-80 for bit descriptions.		
0xYF FA7A		CSOR7	Asynchronous chip select option register 7. See Table 3-81 for bit descriptions.		
0xYF FA7C		BCSBAR	Base address register 7. See Table 3-84 for bit descriptions.		
0xYF FA7E		BCSOR1	Base option register 1. See Table 3-85 for bit descriptions.		

Table A-13 TouCAN (2) (CAN 2.0B Controller)

Address	Access	Symbol	Register	Size	Reset
0xYF FA80	S	TCNMCR	TouCAN Module Configuration Register. See Table 10-13 for bit descriptions.	16	X
0xYF FA82	T	TTR	TouCAN Test Register	16	X
0xYF FA84	S	CANICR	TouCAN Interrupt Configuration Register. See Table 10-14 for bit descriptions.	16	X
0xYF FA86	S	CANCTRL0/ CANCTRL1	TouCAN Control Register 0/ TouCAN Control Register 1. See Table 10-15 and Table 10-18 for bit descriptions.	16	X
0xYF FA88	S	PRESDIV/ CANCTRL2	TouCAN Control and Prescaler Divider Register/TouCAN Control Register 2. See Table 10-19 and Table 10-20 for bit descriptions.	16	X
0xYF FA8A	S	TIMER	TouCAN Free-Running Timer Register. See Table 10-21 for bit descriptions.	16	X
0xYF FA90	S	RXGMASKHI	TouCAN Receive Global Mask High. See Table 10-22 for bit descriptions.	16	X
0xYF FA92	S	RXGMASKLO	TouCAN Receive Global Mask Low. See Table 10-22 for bit descriptions.	16	X
0xYF FA94	S	RX14MASKHI	TouCAN Receive Buffer 14 Mask High.	16	X
0xYF FA96	S	RX14MASKLO	TouCAN Receive Buffer 14 Mask Low.	16	X
0xYF FA98	S	RX15MASKHI	TouCAN Receive Buffer 15 Mask High.	16	X
0xYF FA9A	S	RX15MASKLO	TouCAN Receive Buffer 15 Mask Low.	16	X
0xYF FAA0	S	ESTAT	TouCAN Error and Status Register. See Table 10-23 for bit descriptions.	16	X
0xYF FAA2	S	IMASK	TouCAN Interrupt Masks. See Table 10-26 for bit descriptions.	16	X
0xYF FAA4	S	IFLAG	TouCAN Interrupt Flags. See Table 10-27 for bit descriptions.	16	X
0xYF FAA6	S	RXECTR/ TXECTR	TouCAN Receive Error Counter/ TouCAN Transmit Error Counter. See Table 10-28 for bit descriptions.	16	X

Table A-14 SRAM Modules (2)
(Static Random Access Memory)

SRAM(E)					
0xYF FB00	S	RAMMCR	SRAM4 Module Configuration Register. See **Table 12-1** for bit descriptions.	16	X
0xYF FB02	T	RAMTST	SRAM4 test register.	16	X
0xYF FB04	S	RAMBAH	SRAM4 base address high register. See **Table 12-3** for bit descriptions.	16	X
0xYF FB06	S	RAMBAL	SRAM4 base address low register. See **Table 12-3** for bit descriptions.	16	X

Table A-15 QSM_A (Queued Serial Module A)

0xYF FC00	S	QSMCR_A	QSM Module Configuration Register. See **Table 7-7** for bit descriptions.	16	X
0xYF FC02	T	QTEST_A	QSM Test Register.	16	X
0xYF FC04	S	QILR/QIVR_A	QSM Interrupt Level/interrupt Vector Register. See **Table 7-8** and **Table 7-9** for bit descriptions.	16	X
0xYF FC06	S	—	Reserved	—	X
0xYF FC08	S	SCCR0_A	SCI1 Control Register 0. See **Table 7-22** for bit descriptions.	16	X
0xYF FC0A	S	SCCR1_A	SCI1Control Register 1. See **Table 7-24** for bit descriptions.	16	X
0xYF FC0C	S	SCSR_A	SCI1 Status Register. See **Table 7-26** for bit descriptions.	16	X
0xYF FC0E	S	SCDR_A	SCI1 Data Register. See **Table 7-27** for bit descriptions.	16	X
0xYF FC10 – 0xYF FC12	S	—	Reserved	—	X
0xYF FC14	S	PORTQS_A	QSMCM Port QS Data Register. See **7.2.3.1 QSM Port Data Register (PORTQS)** for bit descriptions.	16	X
0xYF FC16	S	PQSPAR_A / DDRQST_A	QSMCM Port QS Pin Assignment Register/ QSMCM Port QS Data Direction Register. See **Table 7-11** and **Table 7-12** for bit descriptions.	16	X
0xYF FC18	S	SPCR0_A	QSPI Control Register 0. See **Table 7-14** for bit descriptions.	16	X
0xYF FC1A	S	SPCR1_A	QSPI Control Register 1. See **Table 7-16** for bit descriptions.	16	X
0xYF FC1C	S	SPCR2_A	QSPI Control Register 2. See **Table 7-17** for bit descriptions.	16	X
0xYF FC1E	S	SPCR3_A / SPSR_A	QSPI Control Register 3/QSPI Status Register. See **Table 7-18** and **Table 7-19** for bit descriptions.	16	X
0xYF FC20 — 0xYF FDFF	S	—	Reserved	16	X
0xYF FD00 — 0xYF FD1F	S	SCRQ_A	QSPI Receive Queue Locations. See **7.4.4 Receiver Operation** for bit descriptions.	16	X

Table A-15 QSM_A (Queued Serial Module A)

0xYF FD20 — 0xYF FD3F	S	SCTQ_A	QSPI Transmit Queue Locations. See 7.4.3 Transmitter Operation for bit descriptions.	16	X
0xYF FD40 — 0xYF FD4F	S	QSPIRAM_A	QSPI Transmit Queue Locations. See Table 7-20 for bit descriptions.	16	X

Table A-16 TPU3_A (Time Processor Unit)

0xYF FE00	S	TPUMCR_A	TPU3 module configuration register.	16	X
0xYF FE02	T	TCR_A	TPU3 test configuration register	16	X
0xYF FE04	S	DSCR_A	Development support control register.	16	X
0xYF FE06	S	DSSR_A	Development support status register.	16	X
0xYF FE08	S	TICR_A	TPU3 interrupt configuration register.	16	X
0xYF FE0A	S	CIER_A	Channel interrupt enable register.	16	X
0xYF FE0C	S	CFSR0_A	Channel function selection register 0.	16	X
0xYF FE0E	S	CFSR1_A	Channel function selection register 1.	16	X
0xYF FE10	S	CFSR2_A	Channel function selection register 2.	16	X
0xYF FE12	S	CFSR3_A	Channel function selection register 3.	16	X
0xYF FE14	S	HSQR0_A	Host sequence register 0.	16	X
0xYF FE16	S	HSQR1_A	Host sequence register 1.	16	X
0xYF FE18	S	HSRR0_A	Host service request register 0.	16	X
0xYF FE1A	S	HSRR1_A	Host service request register 1.	16	X
0xYF FE1C	S	CPR0_A	Channel priority register 0.	16	X
0xYF FE1E	S	CPR1_A	Channel priority register 1.	16	X
0xYF FE20	S	CISR_A	Channel interrupt status register.	16	X
0xYF FE22	T	LR_A	Link register	16	X
0xYF FE24	T	SGLR_A	Service grant latch register	16	X
0xYF FE26	T	DCNR_A	Decoded channel number register	16	X
0xYF FE28	S	TPUMCR2_A	TPU module configuration register 2.	16	X
0xYF FE2A	S	TPUMCR3_A	TPU module configuration 3.	16	X
0xYF FE2C	T	ISDR_A	Internal scan data register	16	X
0xYF FE2E	T	ISCR_A	Internal scan control register	16	X
0xYF FF00 – 0xYF FF0F	S	—	Channel 0 parameter registers	16	X
0xYF FF10 – 0xYF FF1F	S	—	Channel 1 parameter registers	16	X
0xYF FF20 – 0xYF FF2F	S	—	Channel 2 parameter registers	16	X
0xYF FF30 – 0xYF FF3F	S	—	Channel 3 parameter registers	16	X
0xYF FF40 – 0xYF FF4F	S	—	Channel 4 parameter registers	16	X
0xYF FF50 – 0xYF FF5F	S	—	Channel 5 Parameter Registers	16	X
0xYF FF60 – 0xYF FF6F	S	—	Channel 6 parameter registers	16	X
0xYF FF70 – 0xYF FF7F	S	—	Channel 7 parameter registers	16	X
0xYF FF80 – 0xYF FF8F	S	—	Channel 8 parameter registers	16	X

Table A-16 TPU3_A (Time Processor Unit) (Continued)

0xYF FF90 – 0xYF FF9F	S	—	Channel 9 parameter registers	16	X
0xYF FFA0 – 0xYF FFAF	S	—	Channel 10 parameter registers	16	X
0xYF FFB0 – 0xYF FFBF	S	—	Channel 11 parameter registers	16	X
0xYF FFC0 – 0xYF FFCF	S	—	Channel 12 parameter registers	16	X
0xYF FFD0 – 0xYF FFDF	S	—	Channel 13 parameter registers	16	X
0xYF FFE0 – 0xYF FFEF	S	—	Channel 14 parameter registers	16	X
0xYF FFF0 – 0xYF FFFF	S	—	Channel 15 parameter registers	16	X

APPENDIX D
TPU ROM FUNCTIONS

The following pages provide brief descriptions of the pre-programmed functions in the TPU3. For detailed descriptions, refer to the programming note for the individual function. Please refer to Motorola's ***TPU Literature Package*, TPULITPAK/D**, for the available TPU documentation.

D.1 Overview

The TPU3 contains 4 Kbytes of microcode ROM. This appendix defines the functions that are in the standard ROM on the MC68377. The TPU3 can have up to eight Kbytes of memory and a maximum of four entry tables (see **Figure D-1**).

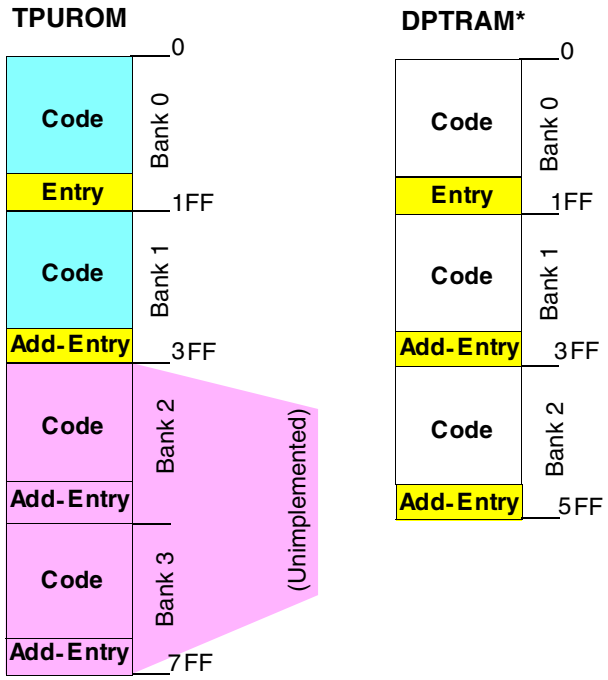


Figure D-1 TPU3 Memory Map

The TPU3 can address up to 8 Kbytes of memory at any one time. It has 4 Kbytes of internal ROM, located in banks zero and one, and 6 Kbytes of dual ported SRAM (DPTRAM), located in banks zero, one and two. As only one type of memory can be used at a time, the TPU3 must either use the internal ROM or the SRAM. Functions from both memory types cannot be used in conjunction.

A new feature of the TPU3 microcode ROM is the existence of two entry tables in the 4 Kbytes of internal ROM. Each entry table has a set of sixteen functions and the user defines which of the two tables the TPU3 will be able to access. Only one table can be used at a time and functions from the two entry tables cannot be mixed. The default entry table is located in bank zero. This table is identical to the standard microcode ROM in the TPU2, so any CPU code written for the TPU2 will work unchanged on the TPU3. The TPU2 and TPU3 ROMs are different than the original TPU ROM. The functions in the default entry table in bank zero are listed in **Table D-1**.

Table D-1 Bank 0 Functions

Function Number	Function Nickname	Function Name
0xF	PTA	Programmable Time Accumulator
0xE	QOM	Queued Output Match
0xD	TSM	Table Stepper Motor
0xC	FQM	Frequency Measurement
0xB	UART	Universal Asynchronous Receiver/Transmitter
0xA	NITC	New Input Capture/Input Transition Counter
9	COMM	Multiphase Motor Commutation
8	HALLD	Hall Effect Decode
7	MCPWM	Multi-Channel Pulse Width Modulation
6	FQD	Fast Quadrature Decode
5	PPWA	Period/Pulse Width Accumulator
4	OC	Output Compare
3	PWM	Pulse Width Modulation
2	DIO	Discrete Input/Output
1	SPWM	Synchronized Pulse Width Modulation
0	SIOP	Serial Input/output Port

The functions in the entry table in bank one are listed in Table 2.

Table D-2 Bank 1 Functions

Function Number	Function Nickname	Function Name
0xF	PTA	Programmable Time Accumulator
0xE	QOM	Queued Output Match
0xD	TSM	Table Stepper Motor
0xC	FQM	Frequency Measurement
0xB	UART	Universal Asynchronous Receiver/Transmitter
0xA	NITC	New Input Capture/Input Transition Counter
9	COMM	Multiphase Motor Commutation
8	HALLD	Hall Effect Decode
7	Reserved	
6	FQD	Fast Quadrature Decode
5	ID	Identification
4	OC	Output Compare
3	PWM	Pulse Width Modulation

Table D-2 Bank 1 Functions (Continued)

Function Number	Function Nickname	Function Name
2	DIO	Discrete Input/Output
1	RWTPIN	Read/Write Timers and Pin
0	SIOP	Serial Input/Output Port

The functions in the bank one entry table are identical to the bank zero entry table functions with three exceptions. Function 1, SPWM, has been replaced by RWTPIN. This is a function that allows the user to read and write to the TPU timebases and corresponding pin. Function 5, PPWA, is now an identification function in the second table. The microcode ROM revision number is provided by this function. Finally, Function 7, MCPWM, has been removed and left open for future use.

The CPU selects which entry table to use by setting the ETBANK field in the TPUMCR2 register. This register is write once after reset. Although one entry table is specified at start-up, it is possible, in some cases, to use functions from both tables without resetting the microcontroller. A customer may, for example, wish to use the ID function from bank one to verify the TPU microcode version but then use the MCPWM function from bank zero. As a customer will typically only run the ID function during system configuration, and not again after that, the bank one entry table can be changed to the bank zero entry table using the soft reset feature of the TPU3. The procedure should be:

1. Set ETBANK field in TPUMCR2 to %01 to select the entry table in bank one
2. Run the ID function
3. Stop the TPU3 by setting the STOP bit in the TPUMCR to one.
4. Reset the TPU3 by setting the SOFTRST bit in the TPUMCR2 register
5. Wait at least nine clocks
6. Clear the SOFTRST bit in the TPUMCR2 register

The TPU3 stays in reset until the CPU clears the SOFTRST bit. After the SOFTRST bit has been cleared the TPU3 will be reset and the entry table in bank zero will be selected by default. To select the bank zero entry table write %00 to the ETBANK field in TPUMCR 2. It is good practice always to initialize any write once register to ensure an incorrect value is not accidentally written.

The descriptions below document the functions listed in **Table D-1** (bank0) and **Table D-2** (bank1) of the TPU3 ROM module.

D.2 Programmable Time Accumulator (PTA)

PTA accumulates a 32-bit sum of the total high time, low time, or period of an input signal over a programmable number of periods or pulses. The period accumulation can start on a rising or falling edge. After the specified number of periods or pulses, the PTA generates an interrupt request.

From one to 255 period measurements can be accumulated before the TPU interrupts the CPU, providing instantaneous or average frequency measurement capability.

Figure D-2 shows all of the host interface areas for the PTA function.

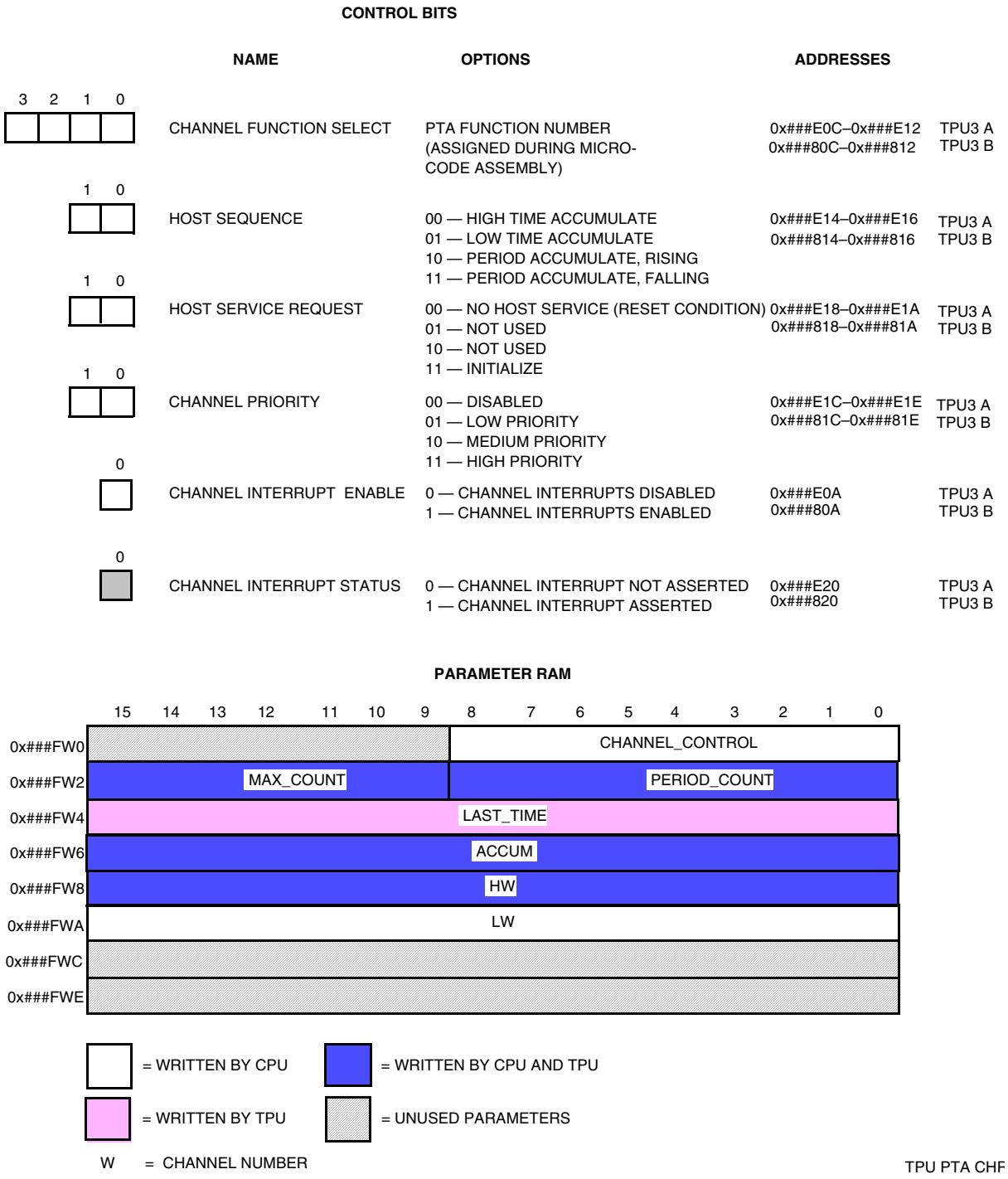


Figure D-2 PTA Parameters

D.3 Queued Output Match TPU Function (QOM)

QOM can generate single or multiple output match events from a table of offsets in parameter RAM. Loop modes allow complex pulse trains to be generated once, a specified number of times, or continuously. The function can be triggered by a link from another TPU channel. In addition, the reference time for the sequence of matches can be obtained from another channel. QOM can generate pulse-width modulated waveforms, including waveforms with high times of 0% or 100%. QOM also allows a TPU channel to be used as a discrete output pin.

Figure D-3 shows all of the host interface areas for the QOM function. The bit encodings shown in **Table D-3** describe the corresponding fields in parameter RAM.

Table D-3 QOM Bit Encoding

A	Timebase Selection
0	Use TCR1 as Timebase
1	Use TCR2 as Timebase

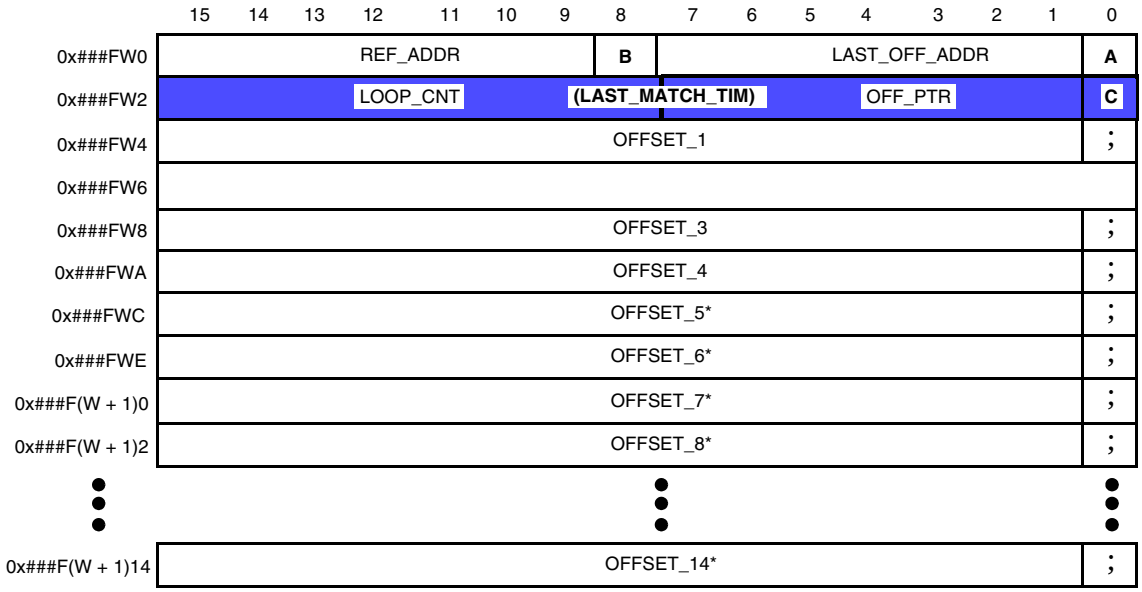
	Edge Selection
0	Falling Edge at Match
1	Rising Edge at Match

B:C	Reference for First Match
00	Immediate TCR Value
01	Last Event Time
10	Value Pointed to by REF_ADDR
11	Last Event Time

CONTROL BITS

3 2 1 0	NAME	OPTIONS	ADDRESSES	
3 2 1 0	CHANNEL FUNCTION SELECT	QOM FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	0x###E0C–0x###E12 0x###80C–0x###812	TPU3 A TPU3 B
1 0	HOST SEQUENCE	00 — SINGLE-SHOT MODE 01 — LOOP MODE 10 — CONTINUOUS MODE 11 — CONTINUOUS MODE	0x###E14–0x###E16 0x###814–0x###816	TPU3 A TPU3 B
1 0	HOST SERVICE REQUEST	00 — NO HOST SERVICE (RESET CON.) 01 — INITIALIZE, NO PIN CHANGE 10 — INITIALIZE, PIN LOW 11 — INITIALIZE, PIN HIGH	0x###E18–0x###E1A 0x###818–0x###81A	TPU3 A TPU3 B
1 0	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	0x###E1C–0x###E1E 0x###81C–0x###81E	TPU3 A TPU3 B
0	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	0x###E0A 0x###80A	TPU3 A TPU3 B
0	CHANNEL INTERRUPT STATUS	0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	0x###E20 0x###820	TPU3 A TPU3 B

PARAMETER RAM



* NOT AVAILABLE ON ALL CHANNELS

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = PRIMARY CHANNEL NUMBER

TPU QOM CHRT

Figure D-3 QOM Parameters

D.4 Table Stepper Motor (TSM)

The TSM function provides for acceleration and deceleration control of a stepper motor with a programmable number of step rates up to 58. TSM uses a table in parameter RAM, rather than an algorithm, to define the stepper motor acceleration profile, allowing the user to fully define the profile. In addition, a slew rate parameter allows fine control of the terminal running speed of the motor independent of the acceleration table. The CPU need only write a desired position, and the TPU accelerates, slews, and decelerates the motor to the required position. Full and half step support is provided for two-phase motors. In addition, a slew rate parameter allows fine control of the terminal running speed of the motor independent of the acceleration table.

Figure D-4 and **Figure D-5** show all of the host interface areas for the TSM function when operating in master and slave mode, respectively.

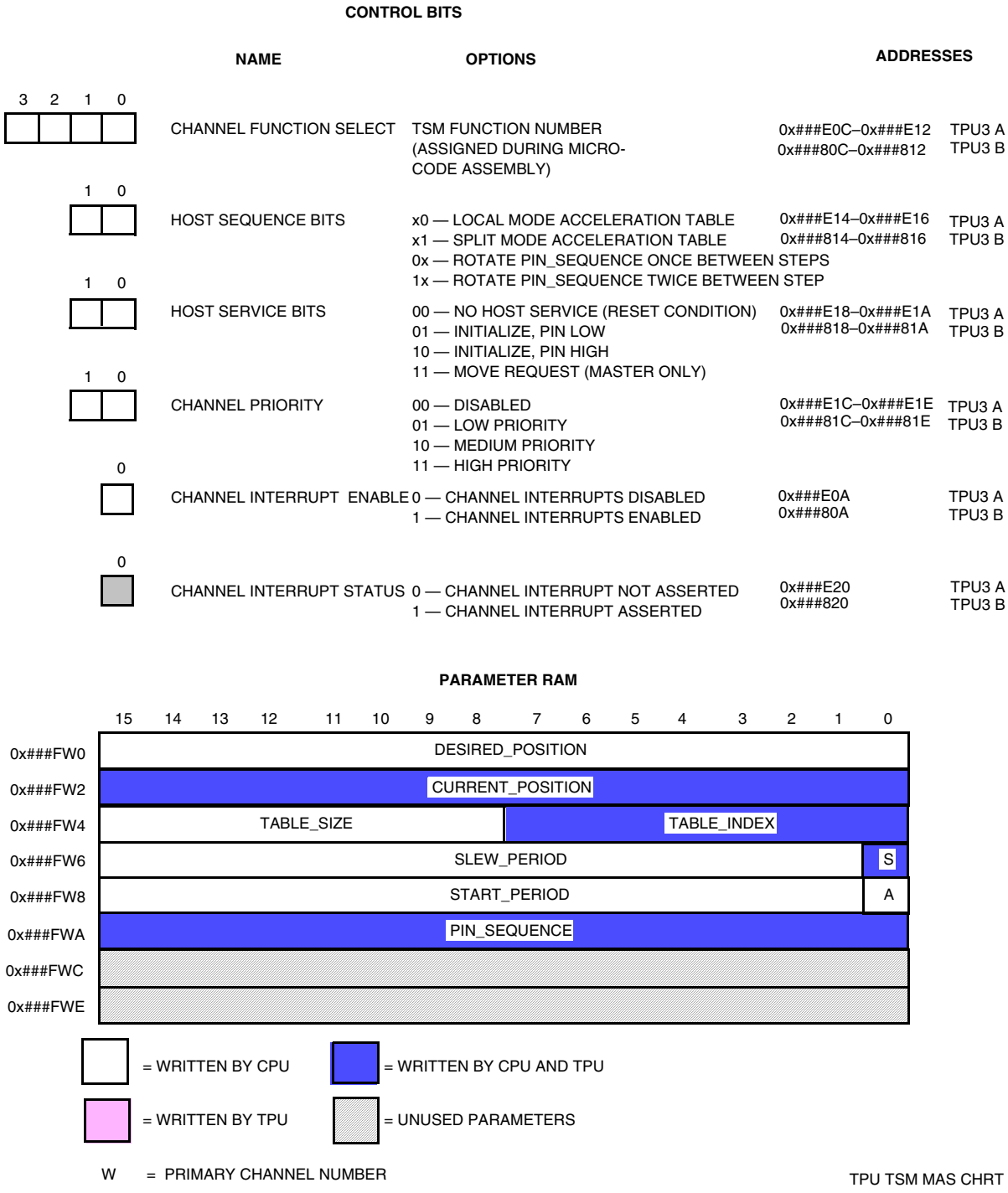


Figure D-4 TSM Parameters — Master Mode

CONTROL BITS

	NAME	OPTIONS	ADDRESSES	
3 2 1 0	CHANNEL FUNCTION SELECT	TSM FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	0x###E0C–0x###E12 0x###80C–0x###812	TPU3 A TPU3 B
1 0	HOST SEQUENCE BITS	x0 — LOCAL MODE ACCELERATION TABLE x1 — SPLIT MODE ACCELERATION TABLE 0x — ROTATE PIN_SEQUENCE ONCE BETWEEN STEPS 1x — ROTATE PIN_SEQUENCE TWICE BETWEEN STEP	0x###E14–0x###E16 0x###814–0x###816	TPU3 A TPU3 B
1 0	HOST SERVICE BITS	00 — NO HOST SERVICE (RESET CON.) 01 — INITIALIZE, PIN LOW 10 — INITIALIZE, PIN HIGH 11 — MOVE REQUEST (MASTER ONLY)	0x###E18–0x###E1A 0x###818–0x###81A	TPU3 A TPU3 B
1 0	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	0x###E1C–0x###E1E 0x###81C–0x###81E	TPU3 A TPU3 B
0	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	0x###E0A 0x###80A	TPU3 A TPU3 B
0	CHANNEL INTERRUPT STATUS	0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	0x###E20 0x###820	TPU3 A TPU3 B

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x###F(W + 1)0	ACCEL_RATIO_2	ACCEL_RATIO_1
0x###F(W + 1)2	ACCEL_RATIO_4	ACCEL_RATIO_3
0x###F(W + 1)4	ACCEL_RATIO_6	ACCEL_RATIO_5
0x###F(W + 1)6	ACCEL_RATIO_8	ACCEL_RATIO_7
0x###F(W + 1)8	ACCEL_RATIO_10	ACCEL_RATIO_9
0x###F(W + 1)A	ACCEL_RATIO_12	ACCEL_RATIO_11
0x###F(W + 1)C	ACCEL_RATIO_14 *	ACCEL_RATIO_13 *
⋮	⋮	⋮
0x###F(W + 3)A	ACCEL_RATIO_36 *	ACCEL_RATIO_35 *

* OPTIONAL ADDITIONAL PARAMETERS NOT AVAILABLE IN ALL CASES. REFER TO MOTOROLA PROGRAMMING NOTE TPUPN04 FOR DETAILS.

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = MASTER CHANNEL NUMBER

TPU TSM SLV

Figure D-5 TSM Parameters — Slave Mode

D.5 Frequency Measurement (FQM)

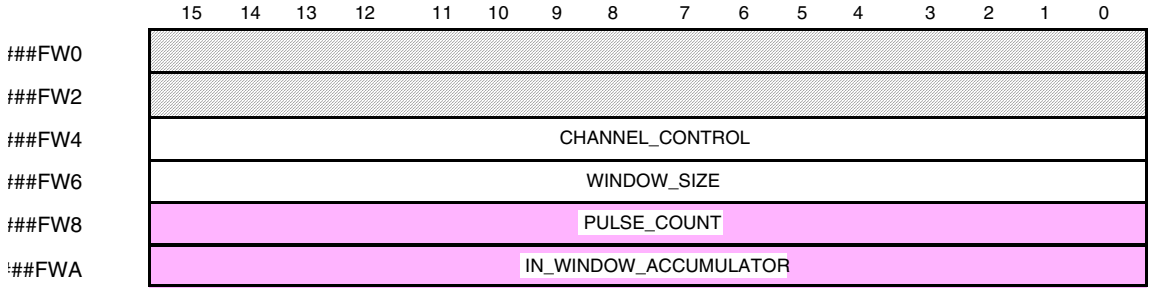
FQM counts the number of input pulses to a TPU channel during a user-defined window period. The function has single shot and continuous modes. No pulses are lost between sample windows in continuous mode. The user selects whether to detect pulses on the rising or falling edge. This function is intended for high speed measurement; measurement of slow pulses with noise rejection can be made with PTA.

Figure D-6 shows all of the host interface areas for the FQM function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES	
0	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	0x###E0C–0x###E12 0x###80C–0x###812	TPU3 A TPU3 B
3 2 1 0	CHANNEL FUNCTION SELECT	xxxx — FQD FUNCTION NUMBER (ASSIGNED DURING MICROCODE ASSEMBLY)	0x###E14–0x###E16 0x###814–0x###816	TPU3 A TPU3 B
1 0	HOST SEQUENCE BITS	00 — BEGIN WITH FALLING EDGE, SINGLE-SHOT MODE 01 — BEGIN WITH FALLING EDGE, CONT. MODE 10 — BEGIN WITH RISING EDGE, SINGLE-SHOT MODE 11 — BEGIN WITH RISING EDGE, CONT. MODE	0x###E18–0x###E1A 0x###818–0x###81A	TPU3 A TPU3 B
1 0	HOST SERVICE BITS	00 — NO HOST SERVICE (RESET CONDITION) 01 — NOT USED 10 — INITIALIZE 11 — NOT USED	0x###E1C–0x###E1E 0x###81C–0x###81E	TPU3 A TPU3 B
1 0	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	0x###E0A 0x###80A	TPU3 A TPU3 B
0	CHANNEL INTERRUPT STATUS	0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	0x###E20 0x###820	TPU3 A TPU3 B

PARAMETER RAM



= WRITTEN BY CPU
= WRITTEN BY CPU AND TPU
= WRITTEN BY TPU
= UNUSED PARAMETERS

W = PRIMARY CHANNEL NUMBER

TPU FQM CHRT

Figure D-6 FQM Parameters

D.6 Universal Asynchronous Receiver/Transmitter (UART)

The UART function uses one or two TPU channels to provide asynchronous communications. Data word length is programmable from one to 14 bits. The function supports detection or generation of even, odd, and no parity. Baud rate is freely programmable and can be higher than 100 Kbaud. Eight bi-directional UART channels running in excess of 9600 baud could be implemented on the TPU.

Figure D-7 and **Figure D-8** show all of the host interface areas for the UART function in transmitting and receiving modes, respectively.

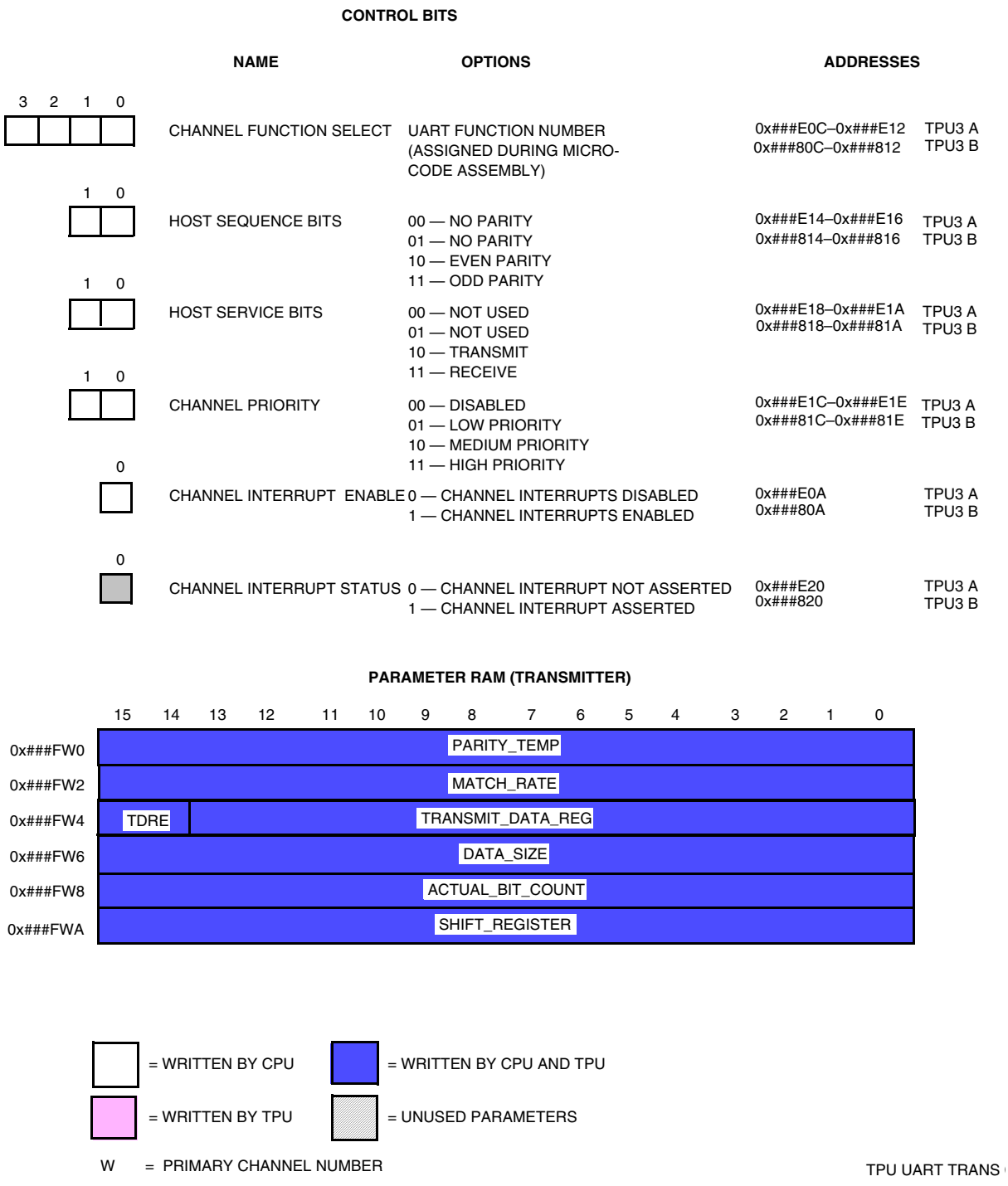


Figure D-7 UART Transmitter Parameters

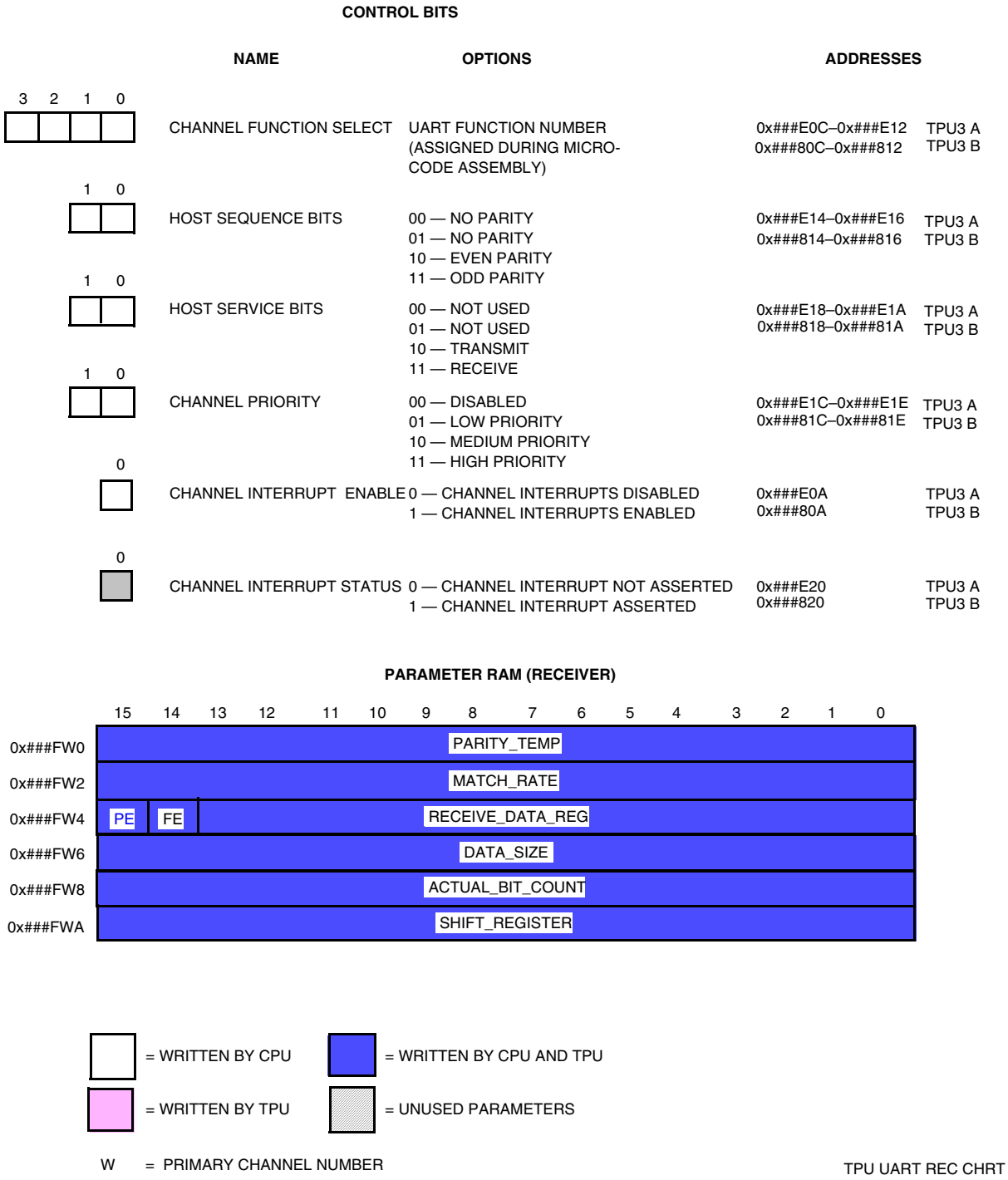


Figure D-8 UART Receiver Parameters

D.7 New Input Capture/Transition Counter (NITC)

Any channel of the TPU can capture the value of a specified TCR or any specified location in parameter RAM upon the occurrence of each transition or specified number of transitions, and then generate an interrupt request to notify the bus master. The times of the most recent two transitions are maintained in parameter RAM. A channel can perform input captures continually, or a channel can detect a single transition or specified number of transitions, ceasing channel activity until reinitialization. After each transition or specified number of transitions, the channel can generate a link to other channels.

Figure D-9 shows all of the host interface areas for the NITC function.

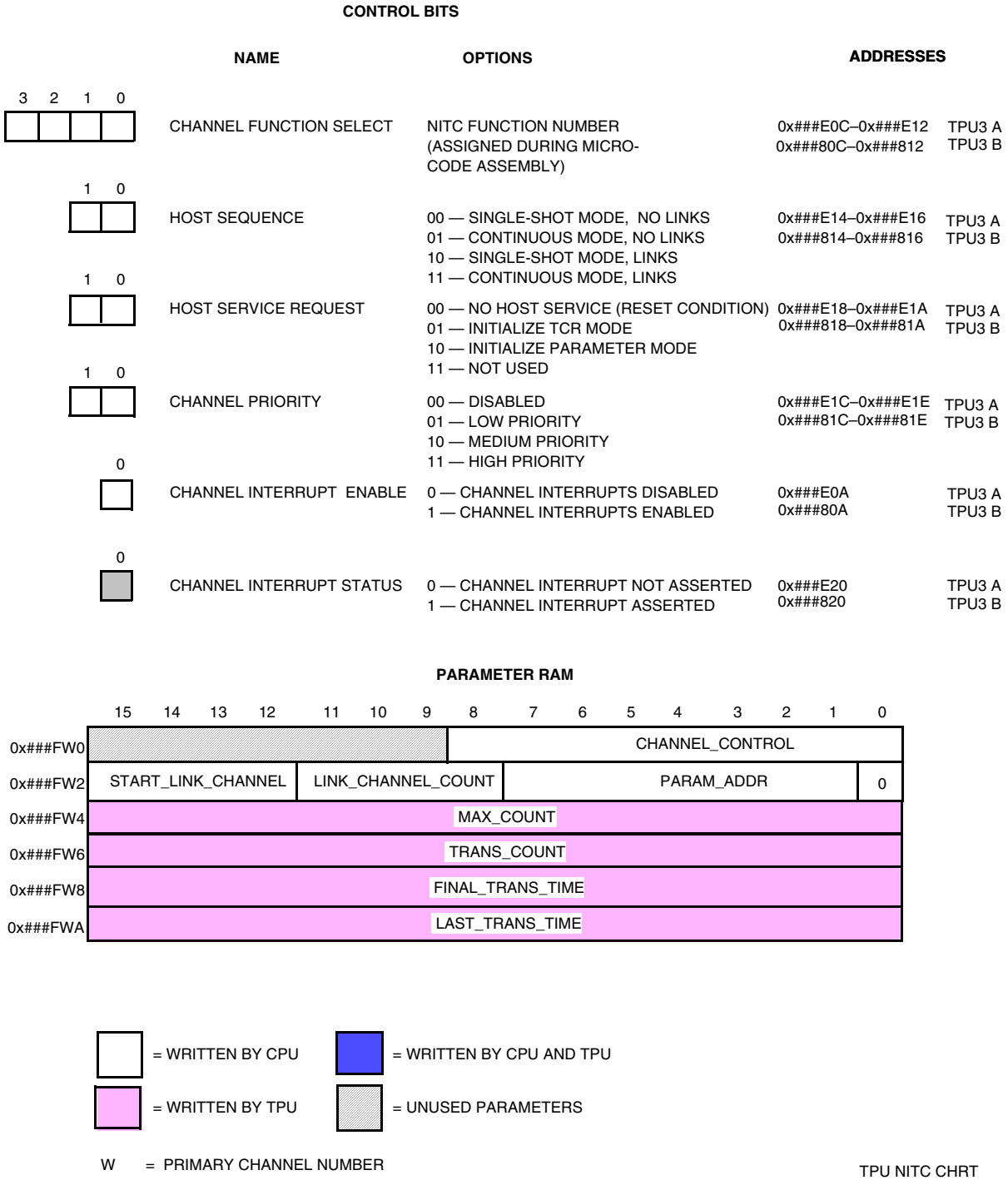


Figure D-9 NITC Parameters

D.8 Multiphase Motor Commutation (COMM)

The COMM function generates the phase commutation signals for a variety of brushless motors, including three-phase brushless direct current. It derives the commutation state directly from the position decoded in FQD, thus eliminating the need for hall effect sensors.

The state sequence is implemented as a user-configurable state machine, thus providing a flexible approach with other general applications. A CPU offset parameter is provided to allow all the switching angles to be advanced or retarded on the fly by the CPU. This feature is useful for torque maintenance at high speeds.

Figure D-10 and **Figure D-11** show all of the host interface areas for the COMM function.

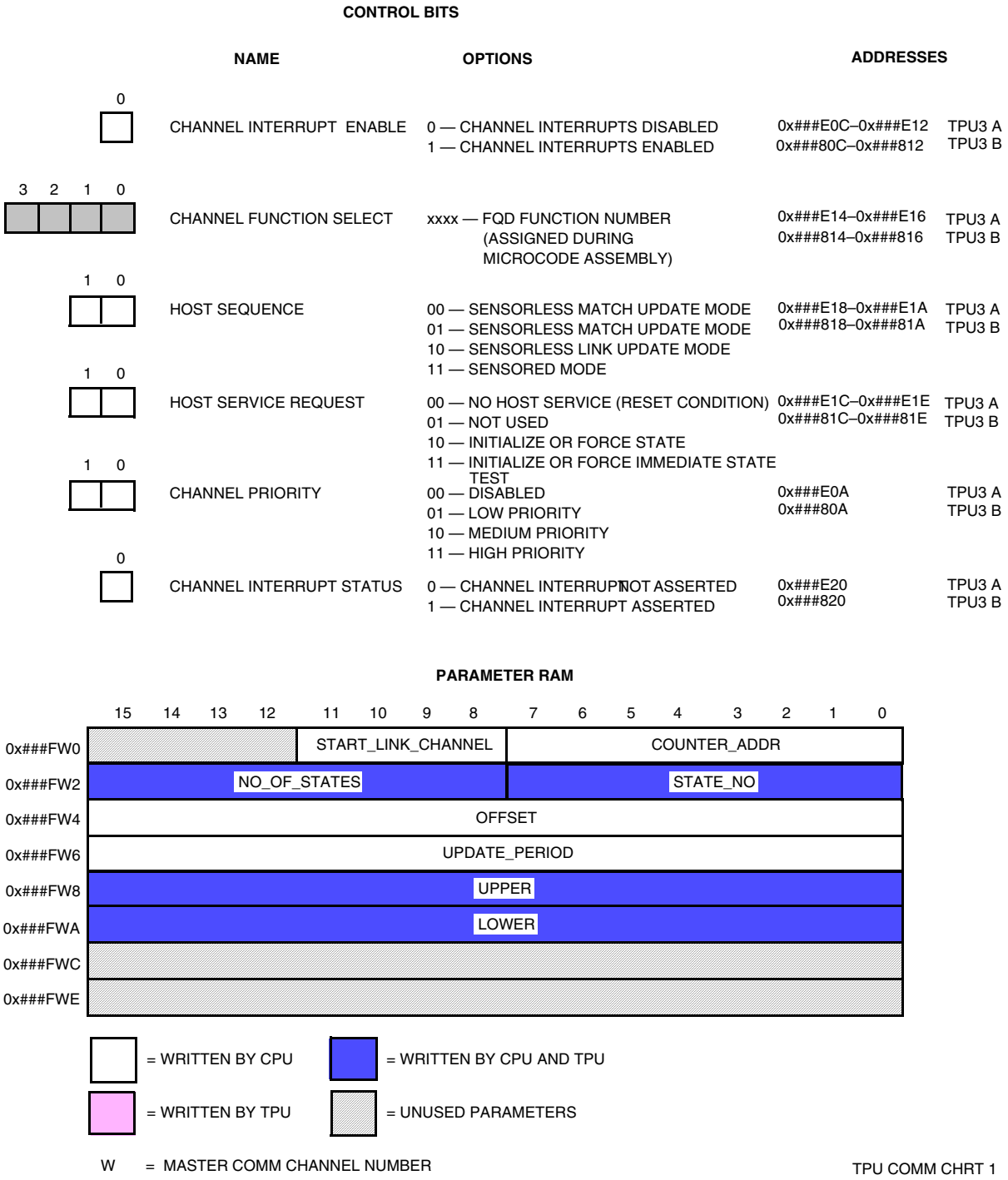


Figure D-10 COMM Parameters, Part 1 of 2

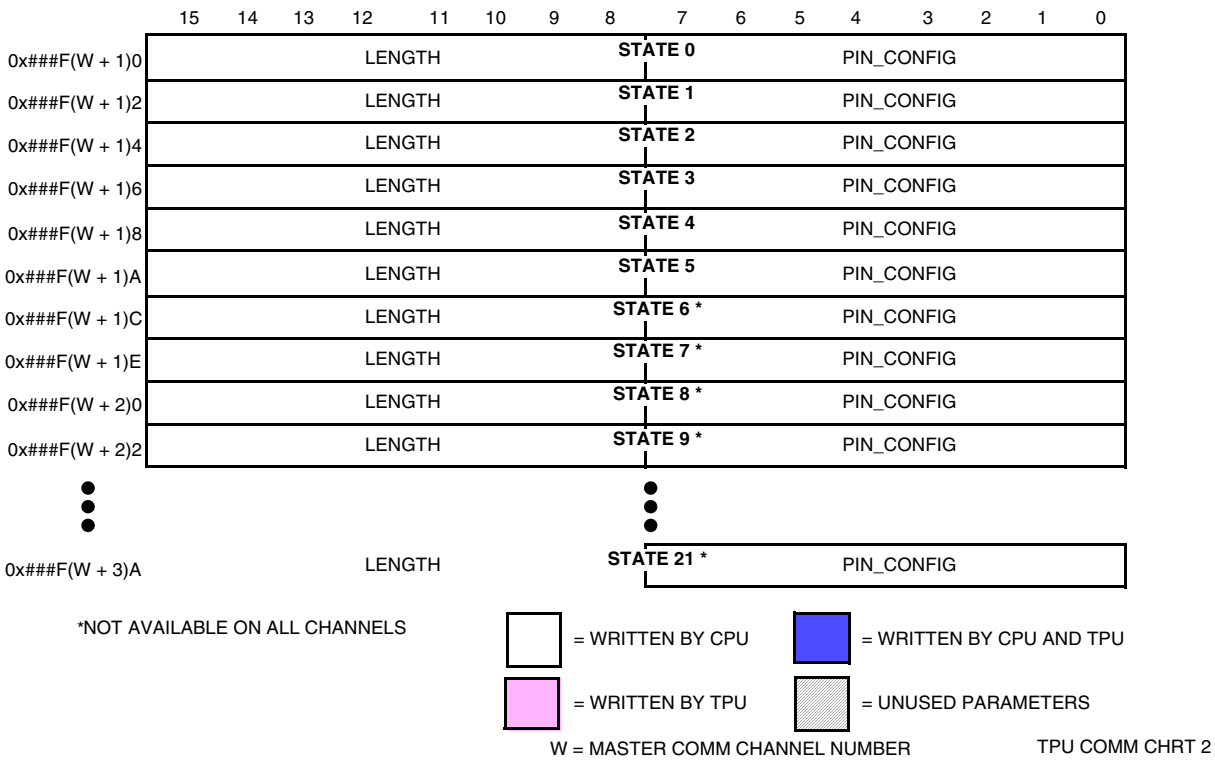


Figure D-11 COMM Parameters, Part 2 of 2

D.9 Hall Effect Decode (HALLD)

The HALLD function decodes the sensor signals from a brushless motor, along with a direction input from the CPU, into a state number. The function supports two- or three-sensor decoding. The decoded state number is written into a COMM channel, which outputs the required commutation drive signals. In addition to brushless motor applications, the function can have more general applications, such as decoding “option” switches.

Figure D-12 shows all of the host interface areas for the HALLD function.

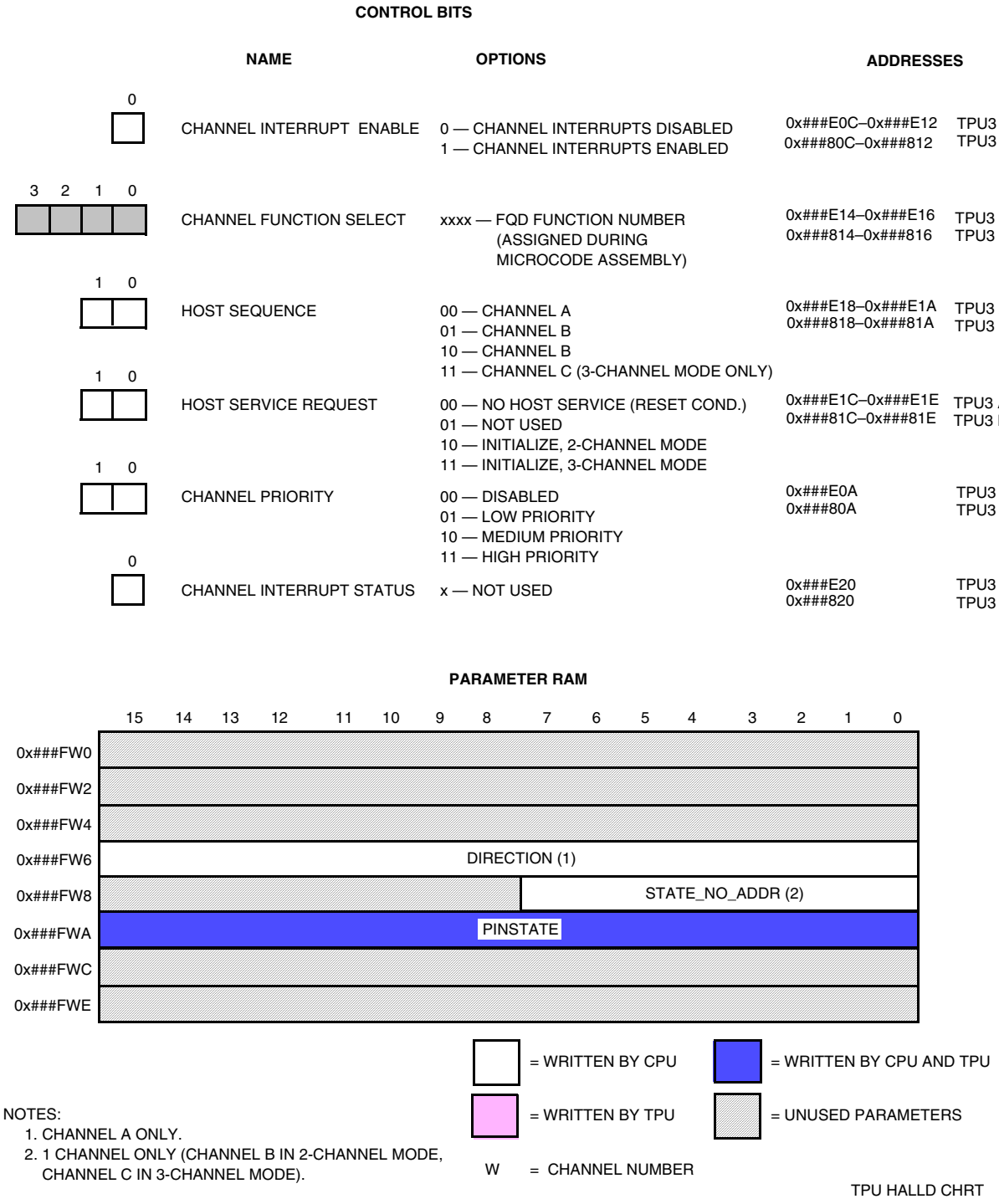


Figure D-12 HALLD Parameters

D.10 Multichannel Pulse-Width Modulation (MCPWM)

MCPWM generates pulse-width modulated outputs with full 0% to 100% duty cycle range independent of other TPU activity. This capability requires two TPU channels plus an external gate for one PWM channel. (A simple one-channel PWM capability is supported by the QOM function.)

Multiple PWMs generated by MCPWM have two types of high time alignment: edge aligned and center aligned. Edge-aligned mode uses n + 1 TPU channels for n PWMs; center-aligned mode uses 2n + 1 channels. Center-aligned mode allows a user-defined “dead time” to be specified so that two PWMs can be used to drive an H-bridge without destructive current spikes. This feature is important for motor control applications.

Figure D-13 through **Figure D-18** show the host interface areas for the MCPWM function in each mode.

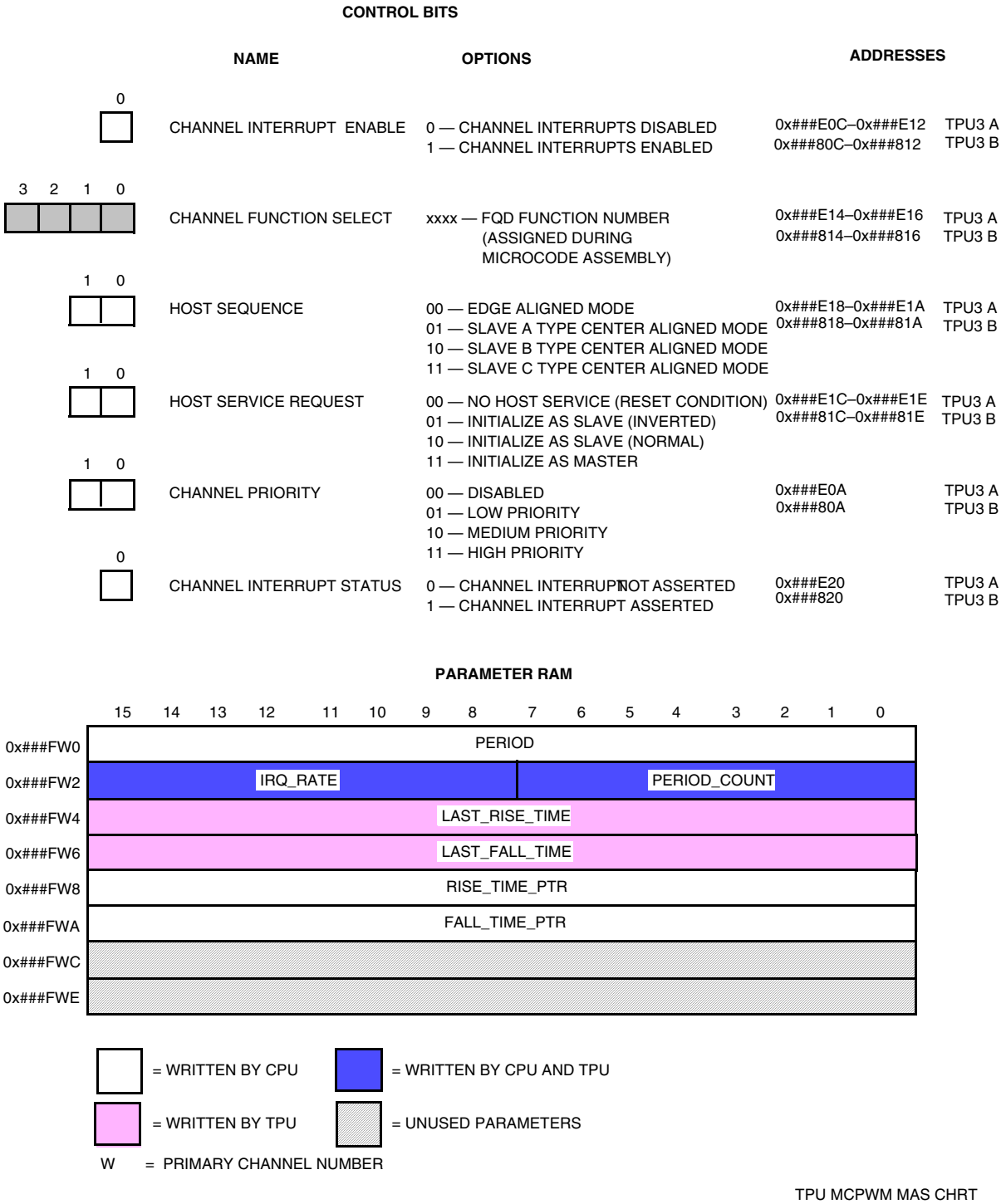


Figure D-13 MCPWM Parameters — Master Mode

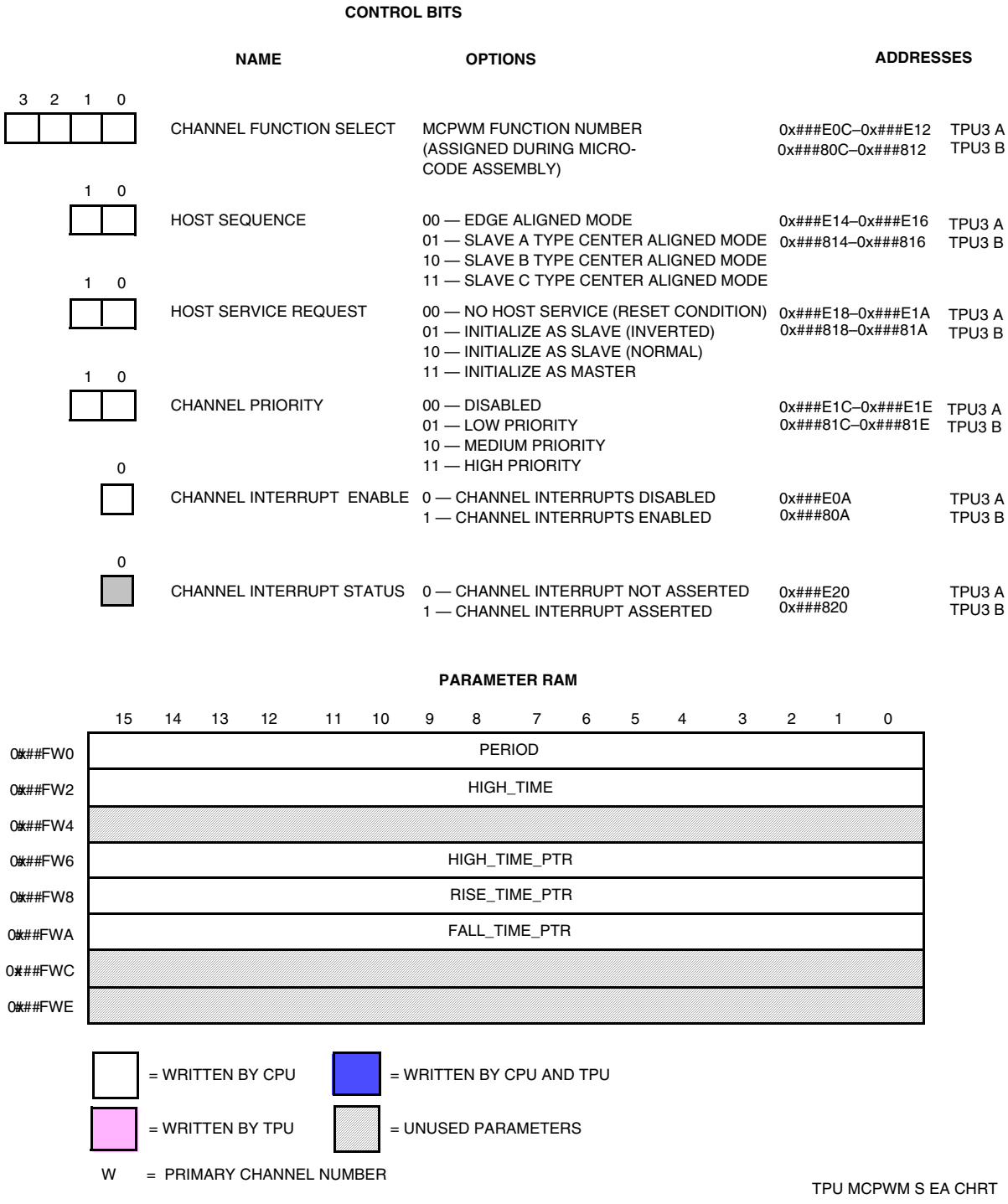


Figure D-14 MCPWM Parameters — Slave Edge-Aligned Mode

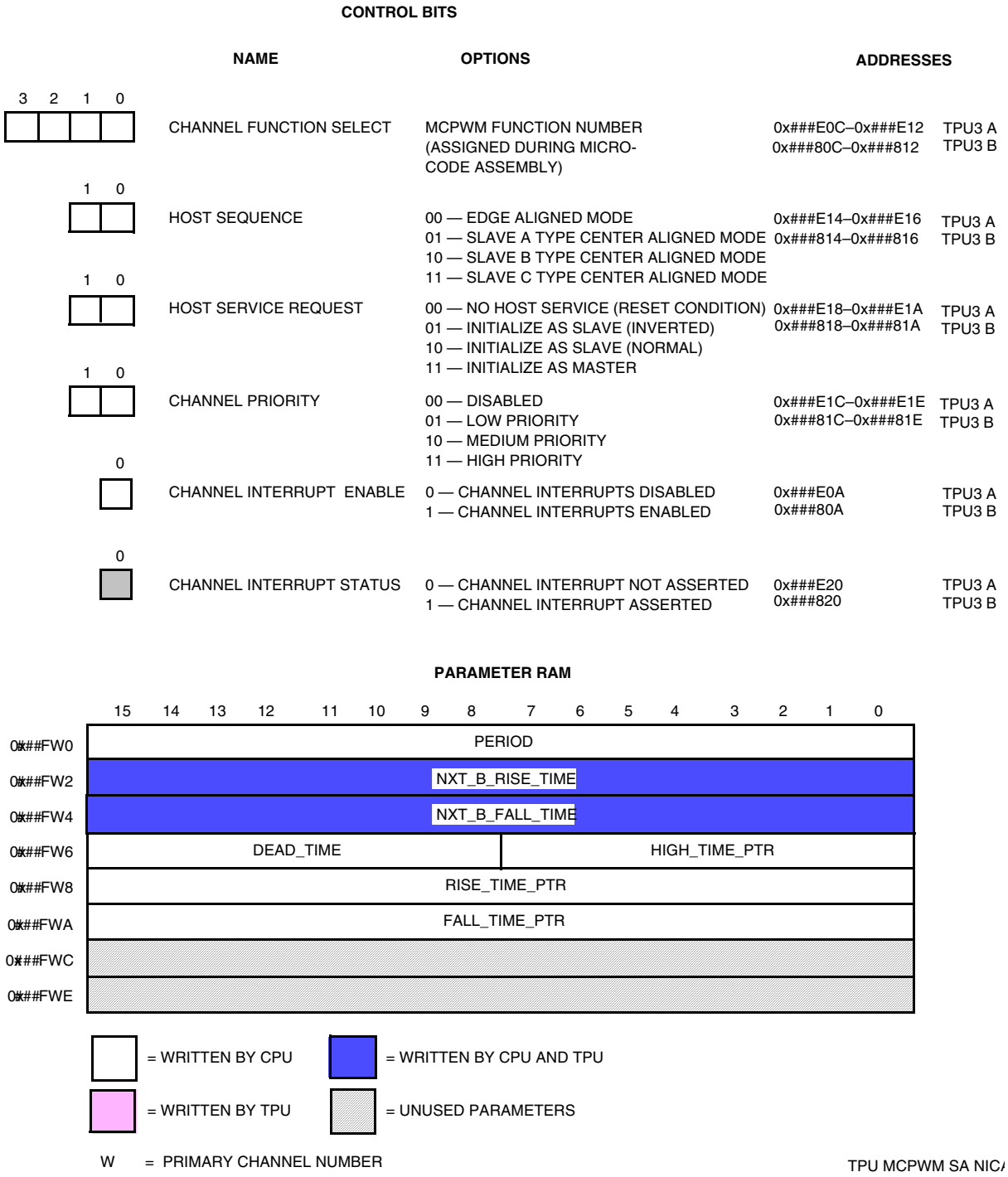


Figure D-15 MCPWM Parameters — Slave Ch A Non-Inverted Center-Aligned Mode

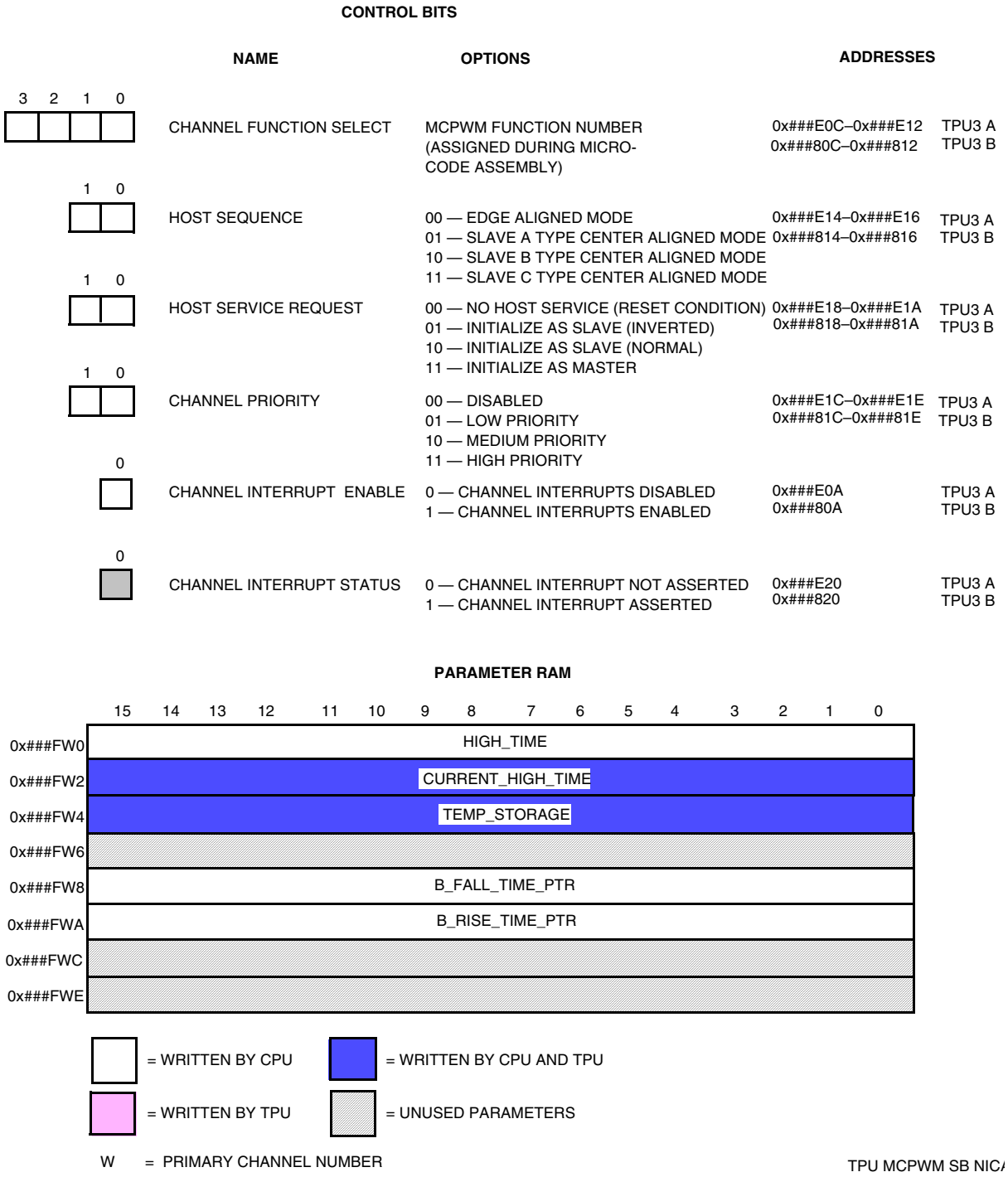


Figure D-16 MCPWM Parameters — Slave Ch B Non-Inverted Center-Aligned Mode

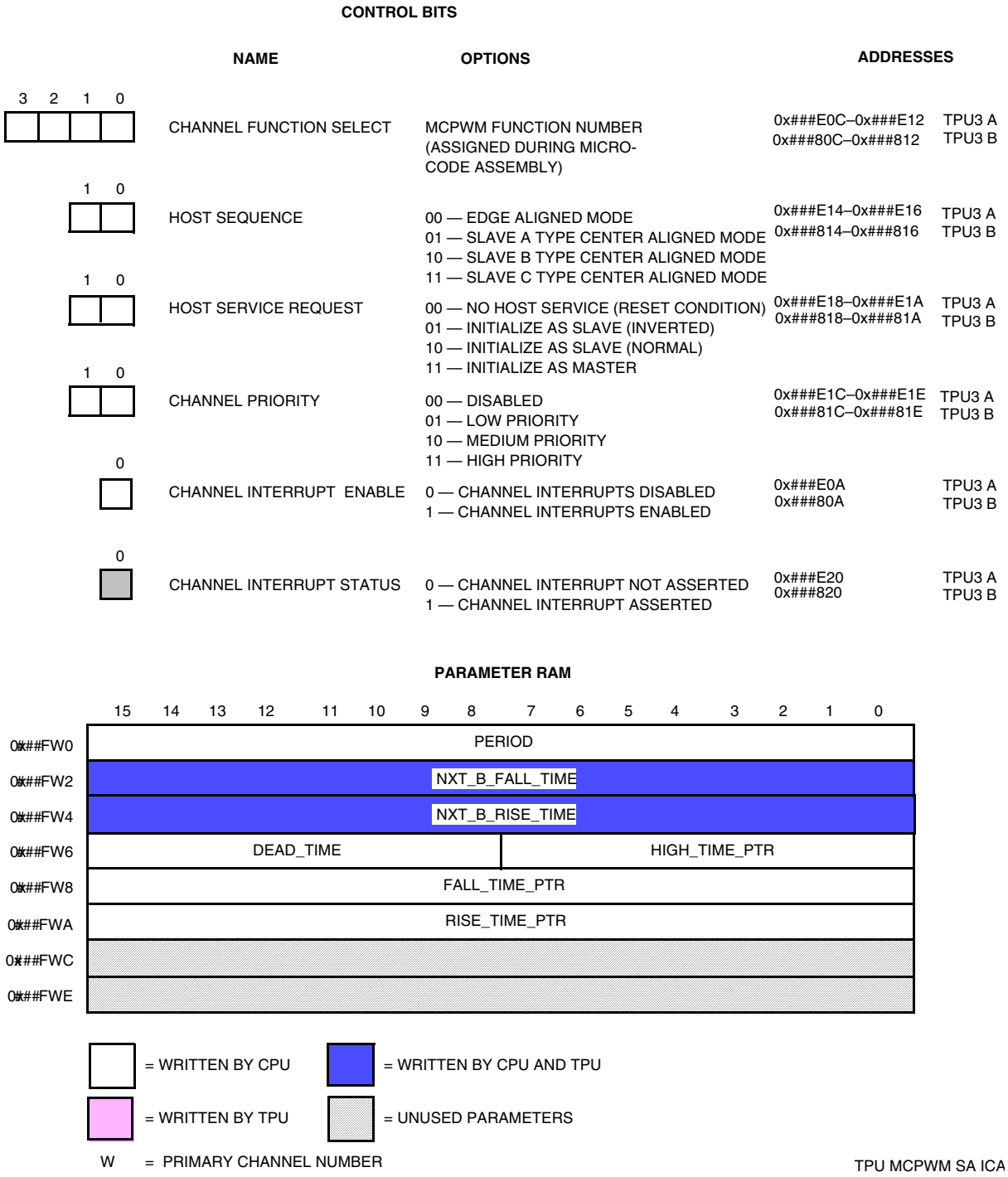


Figure D-17 MCPWM Parameters — Slave Ch A Inverted Center-Aligned Mode

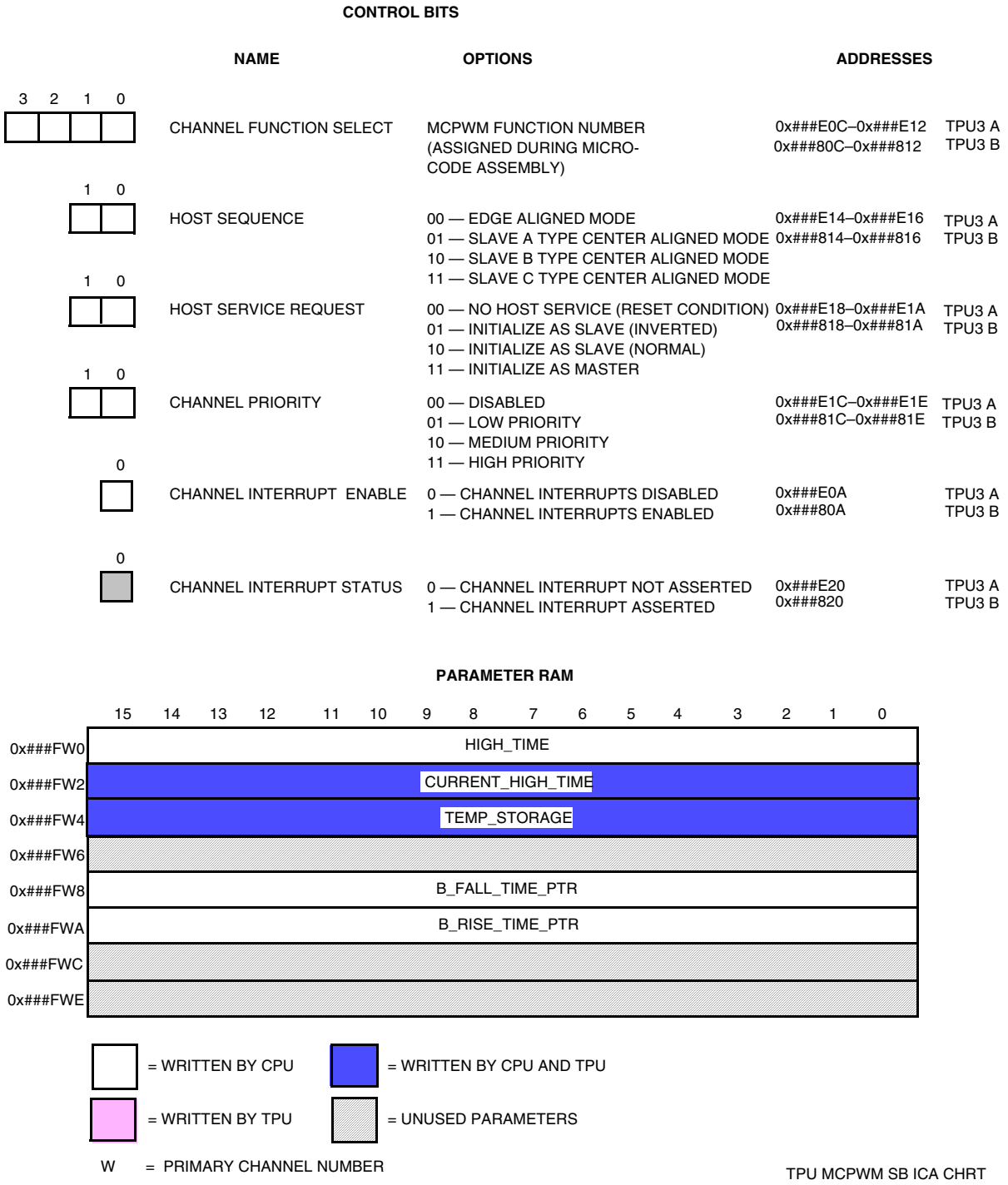


Figure D-18 MCPWM Parameters — Slave Ch B Inverted Center-Aligned Mode

D.11 Fast Quadrature Decode TPU Function (FQD)

FQD is a position feedback function for motor control. It decodes the two signals from a slotted encoder to provide the CPU with a 16-bit free-running position counter. FQD incorporates a “speed switch” which disables one of the channels at high speed, allowing faster signals to be decoded. A time stamp is provided on every counter update to allow position interpolation and better velocity determination at low speed or when low resolution encoders are used. The third index channel provided by some encoders is handled by the ITC function.

Figure D-19 and **Figure D-20** show the host interface areas for the FQD function for primary and secondary channels, respectively.

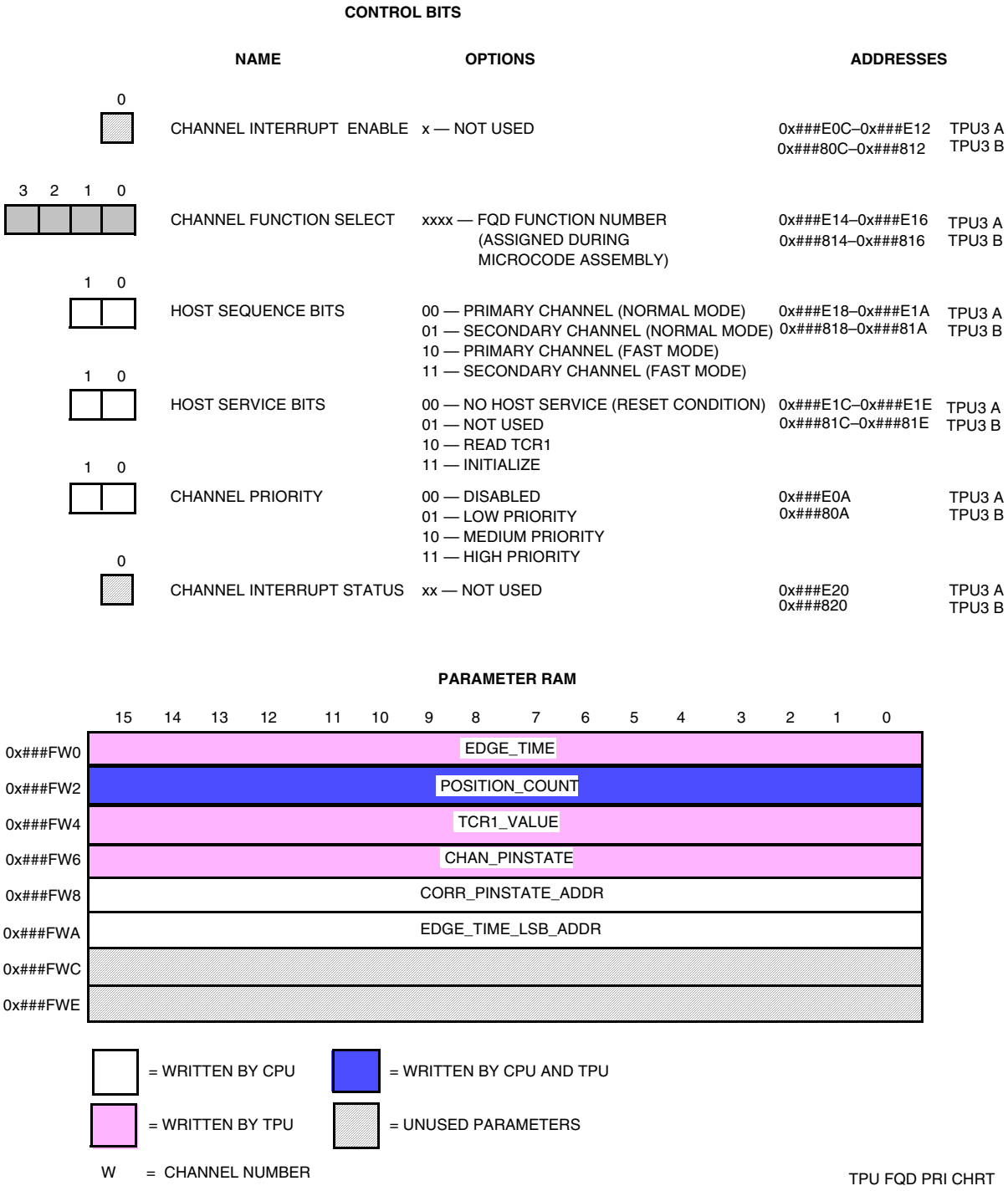


Figure D-19 FQD Parameters — Primary Channel

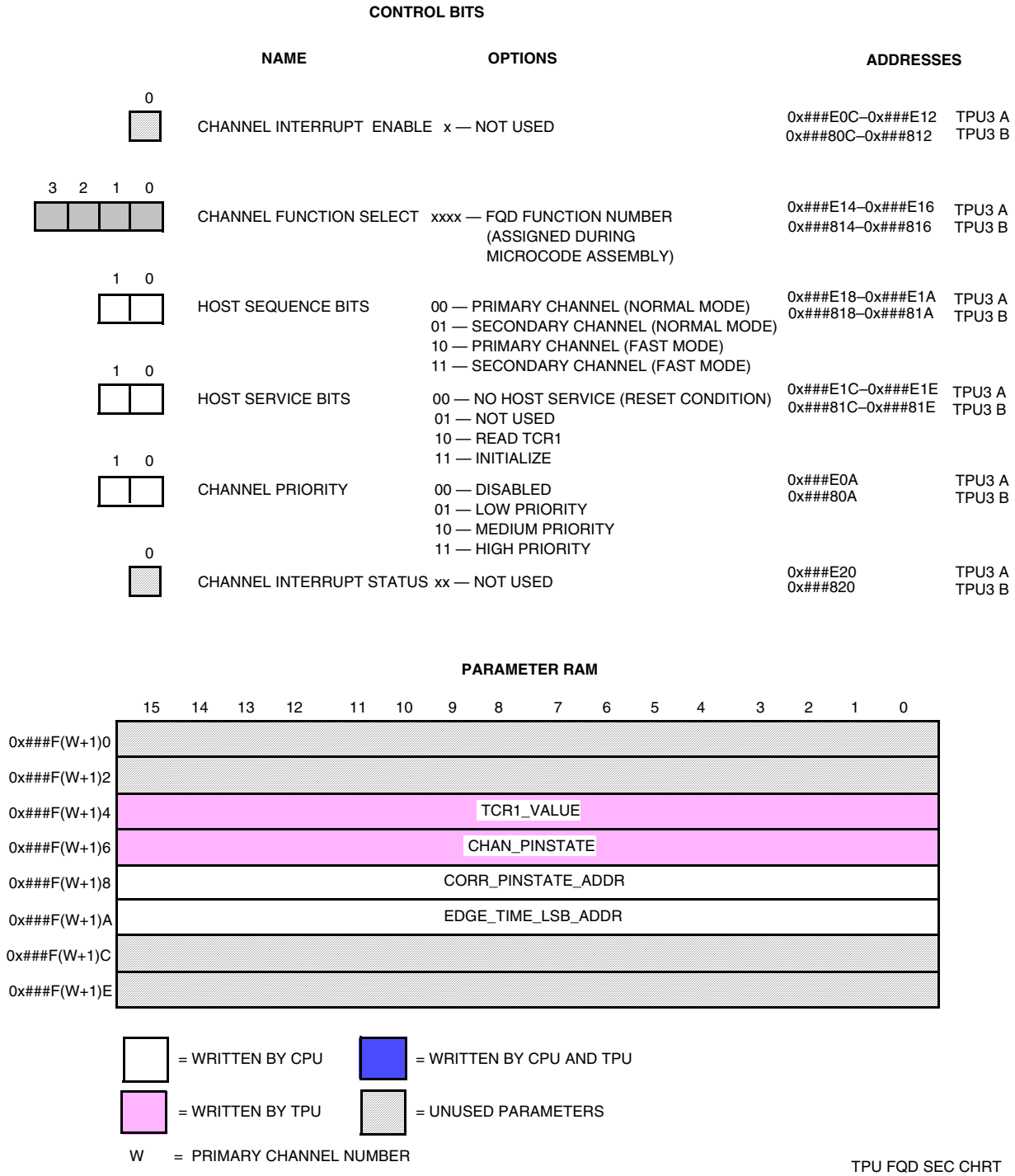


Figure D-20 FQD Parameters — Secondary Channel

D.12 Period/Pulse-Width Accumulator (PPWA)

The period/pulse-width accumulator (PPWA) algorithm accumulates a 16-bit or 24-bit sum of either the period or the pulse width of an input signal over a programmable number of periods or pulses (from one to 255). After an accumulation period, the algorithm can generate a link to a sequential block of up to eight channels. The user specifies a starting channel of the block and number of channels within the block. Generation of links depends on the mode of operation.

Any channel can be used to measure an accumulated number of periods of an input signal. A maximum of 24 bits can be used for the accumulation parameter. From one to 255 period measurements can be made and summed with the previous measurement(s) before the TPU interrupts the CPU, allowing instantaneous or average frequency measurement, and the latest complete accumulation (over the programmed number of periods).

The pulse width (high-time portion) of an input signal can be measured (up to 24 bits) and added to a previous measurement over a programmable number of periods (one to 255). This provides an instantaneous or average pulse-width measurement capability, allowing the latest complete accumulation (over the specified number of periods) to always be available in a parameter.

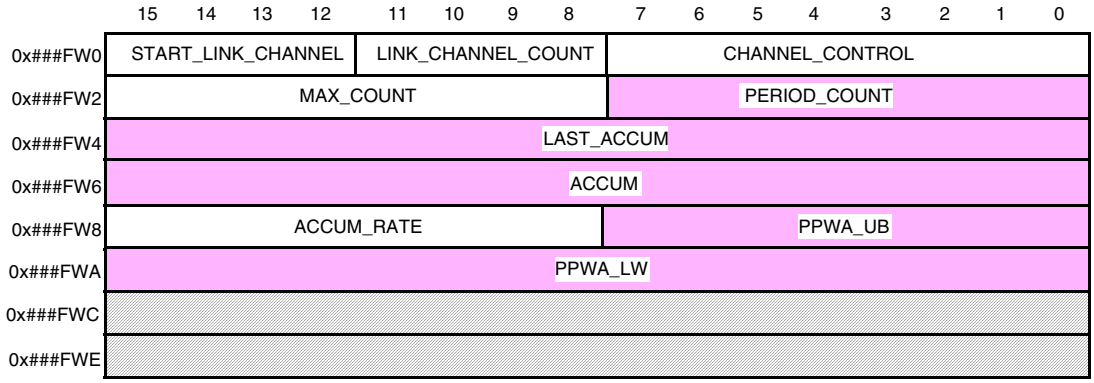
By using the output compare function in conjunction with PPWA, an output signal can be generated that is proportional to a specified input signal. The ratio of the input and output frequency is programmable. One or more output signals with different frequencies, yet proportional and synchronized to a single input signal, can be generated on separate channels.

Figure D-21 shows the host interface areas and parameter RAM for the PPWA function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES	
3 2 1 0	CHANNEL FUNCTION SELECT	PPWA FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	0x###E0C–0x###E12 0x###80C–0x###812	TPU3 A TPU3 B
1 0	CHANNEL PRIORITY	00 — CHANNEL DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	0x###E14–0x###E16 0x###814–0x###816	TPU3 A TPU3 B
1 0	HOST SEQUENCE BITS	00 — ACCUMULATE 24-BIT PERIODS, NO LINKS 01 — ACCUMULATE 16-BIT PERIODS, LINKS 10 — ACCUMULATE 24-BIT PULSE WIDTHS, NO LINKS 11 — ACCUMULATE 16-BIT PULSE WIDTHS, LINKS	0x###E18–0x###E1A 0x###818–0x###81A	TPU3 A TPU3 B
1 0	HOST SERVICE BITS	00 — NOT USED 01 — NOT USED 10 — INITIALIZE 11 — NOT USED	0x###E1C–0x###E1E 0x###81C–0x###81E	TPU3 A TPU3 B
0	INTERRUPT ENABLE	0 — INTERRUPT NOT ASSERTED 1 — INTERRUPT ASSERTED	0x###E0A 0x###80A	TPU3 A TPU3 B
0	INTERRUPT STATUS		0x###E20 0x###820	TPU3 A TPU3 B

PARAMETER RAM



= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = CHANNEL NUMBER

NOTES:
1. THE TPU DOES NOT CHECK THE VALUE OF LINK_CHANNEL_COUNT. IF THIS PARAMETER IS **NOT** > 0 AND ≤ 8, RESULTS ARE UNPREDICTABLE.
2. MAX_COUNT MAY BE WRITTEN AT ANY TIME BY THE HOST CPU, BUT IF THE VALUE WRITTEN IS ≤ PERIOD_COUNT, A PERIOD OR PULSE-WIDTH ACCUMULATION IS TERMINATED. IF THIS HAPPENS, THE NUMBER OF PERIODS OVER WHICH THE ACCUMULATION IS DONE WILL NOT CORRESPOND TO MAX_COUNT.

1049A

Figure D-21 PPWA Parameters

D.13 Output Compare (OC)

The output compare (OC) function generates a rising edge, falling edge, or a toggle of the previous edge in one of three ways:

1. Immediately upon CPU initiation, thereby generating a pulse with a length equal to a programmable delay time
2. At a programmable delay time from a user-specified time
3. Continuously. Upon receiving a link from a channel, OC references, without CPU interaction, a specifiable period and calculates an offset:

$$\text{Offset} = \text{Period} \times \text{Ratio}$$

where RATIO is a parameter supplied by the user.

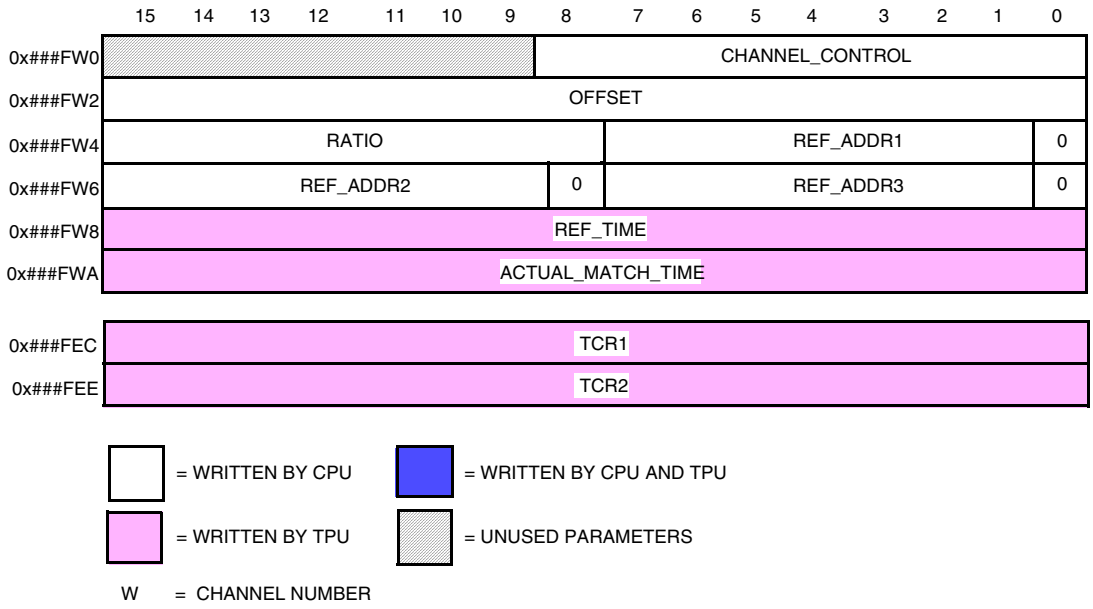
This algorithm generates a 50% duty-cycle continuous square wave with each high/low time equal to the calculated OFFSET. Due to offset calculation, there is an initial link time before continuous pulse generation begins.

Figure D-22 shows the host interface areas and parameter RAM for the OC function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES	
3 2 1 0	CHANNEL FUNCTION SELECT	OC FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	0x###E0C–0x###E12 0x###80C–0x###812	TPU3 A TPU3 B
1 0	CHANNEL PRIORITY	00 — CHANNEL DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	0x###E14–0x###E16 0x###814–0x###816	TPU3 A TPU3 B
1 0	HOST SEQUENCE BITS	0x — MATCHES AND PULSES SCHEDULED 1x — ONLY READ TCR1, TCR2	0x###E18–0x###E1A 0x###818–0x###81A	TPU3 A TPU3 B
1 0	HOST SERVICE BITS	00 — NO HOST SERVICE REQUEST 01 — HOST-INITIATED PULSE 10 — NOT USED 11 — INITIALIZE, CONTINUOUS PULSES	0x###E1C–0x###E1E 0x###81C–0x###81E	TPU3 A TPU3 B
0	INTERRUPT ENABLE	0 — INTERRUPT NOT ASSERTED 1 — INTERRUPT ASSERTED	0x###E0A 0x###80A	TPU3 A TPU3 B
0	INTERRUPT STATUS		0x###E20 0x###820	TPU3 A TPU3 B

PARAMETER RAM



1024A

Figure D-22 OC Parameters

D.14 Pulse-Width Modulation (PWM)

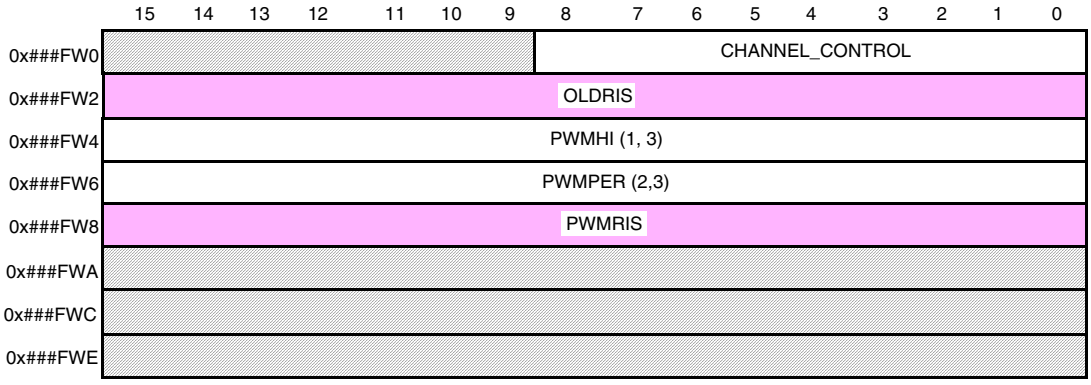
The TPU can generate a pulse-width modulation (PWM) waveform with any duty cycle from zero to 100% (within the resolution and latency capability of the TPU). To define the PWM, the CPU provides one parameter that indicates the period and another parameter that indicates the high time. Updates to one or both of these parameters can direct the waveform change to take effect immediately, or coherently beginning at the next low-to-high transition of the pin.

Figure D-23 shows the host interface areas and parameter RAM for the PWM function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES	
3 2 1 0	CHANNEL FUNCTION SELECT	PWM FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	0x###E0C–0x###E12 0x###80C–0x###812	TPU3 A TPU3 B
1 0	CHANNEL PRIORITY	00 — DISABLE 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	0x###E14–0x###E16 0x###814–0x###816	TPU3 A TPU3 B
1 0	HOST SEQUENCE BITS	xx — NOT USED	0x###E18–0x###E1A 0x###818–0x###81A	TPU3 A TPU3 B
1 0	HOST SERVICE BITS	00 — NOT USED 01 — IMMEDIATE UPDATE OF PWM 10 — INITIALIZE 11 — NOT USED	0x###E1C–0x###E1E 0x###81C–0x###81E	TPU3 A TPU3 B
0	INTERRUPT ENABLE	0 — INTERRUPT NOT ASSERTED 1 — INTERRUPT ASSERTED	0x###E0A 0x###80A	TPU3 A TPU3 B
0	INTERRUPT STATUS		0x###E20 0x###820	TPU3 A TPU3 B

PARAMETER RAM



= WRITTEN BY CPU
= WRITTEN BY CPU AND TPU
= WRITTEN BY TPU
= UNUSED PARAMETERS
W = CHANNEL NUMBER

NOTES:
1. BEST-CASE MINIMUM FOR PWMHI IS 32 SYSTEM CLOCK CYCLES.
2. BEST-CASE MINIMUM FOR PWMPER IS 48 SYSTEM CLOCK CYCLES.
3. PWMHI AND PWMPER MUST BE ACCESSED COHERENTLY.

1027A

Figure D-23 PWM Parameters

D.15 Discrete Input/Output (DIO)

The DIO function allows a TPU channel to be used as a digital I/O pin.

When a pin is used as a discrete input, a parameter indicates the current input level and the previous 15 levels of a pin. Bit 15, the most significant bit of the parameter, indicates the most recent state. Bit 14 indicates the next most recent state, and so on. The programmer can choose one of the three following conditions to update the parameter:

1. when a transition occurs
2. when the CPU makes a request, or
3. when a rate specified in another parameter is matched

When a pin is used as a discrete output, it is set high or low only upon request by the CPU.

Figure D-24 shows the host interface areas for the DIO function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES	
3 2 1 0	CHANNEL FUNCTION SELECT	DIO FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	0x###E0C–0x###E12 0x###80C–0x###812	TPU3 A TPU3 B
1 0	CHANNEL PRIORITY	00 — DISABLE 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	0x###E14–0x###E16 0x###814–0x###816	TPU3 A TPU3 B
1 0	HOST SEQUENCE BITS	00 — UPDATE ON TRANSITION 01 — UPDATE AT MATCH RATE 10 — UPDATE ON HSR 11 11 — NOT USED	0x###E18–0x###E1A 0x###818–0x###81A	TPU3 A TPU3 B
1 0	HOST SERVICE BITS	00 — NOT USED 01 — DRIVE PIN HIGH 10 — DRIVE PIN LOW 11 — INITIALIZE	0x###E1C–0x###E1E 0x###81C–0x###81E	TPU3 A TPU3 B
0	INTERRUPT ENABLE	0 — INTERRUPT NOT ASSERTED 1 — INTERRUPT ASSERTED	0x###E0A 0x###80A	TPU3 A TPU3 B
0	INTERRUPT STATUS		0x###E20 0x###820	TPU3 A TPU3 B

PARAMETER RAM

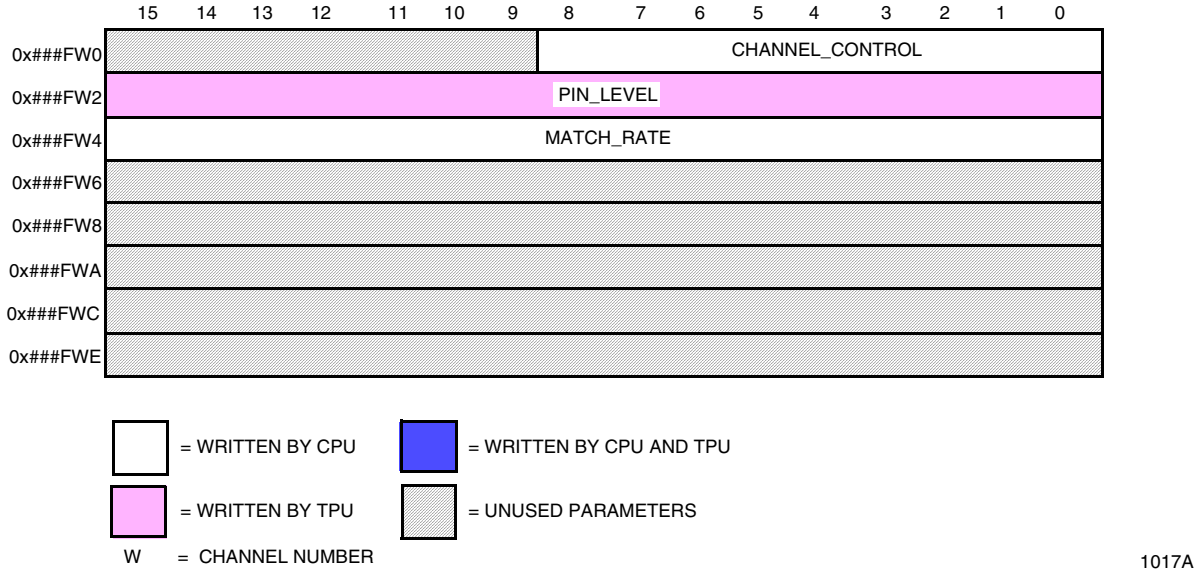


Figure D-24 DIO Parameters

D.16 Synchronized Pulse-Width Modulation (SPWM)

The SPWM function generates a pulse-width modulated waveform (PWM). The CPU can change the period or high time of the waveform at any time. Three different operating modes allow the function to maintain complex timing relationships between channels without CPU intervention.

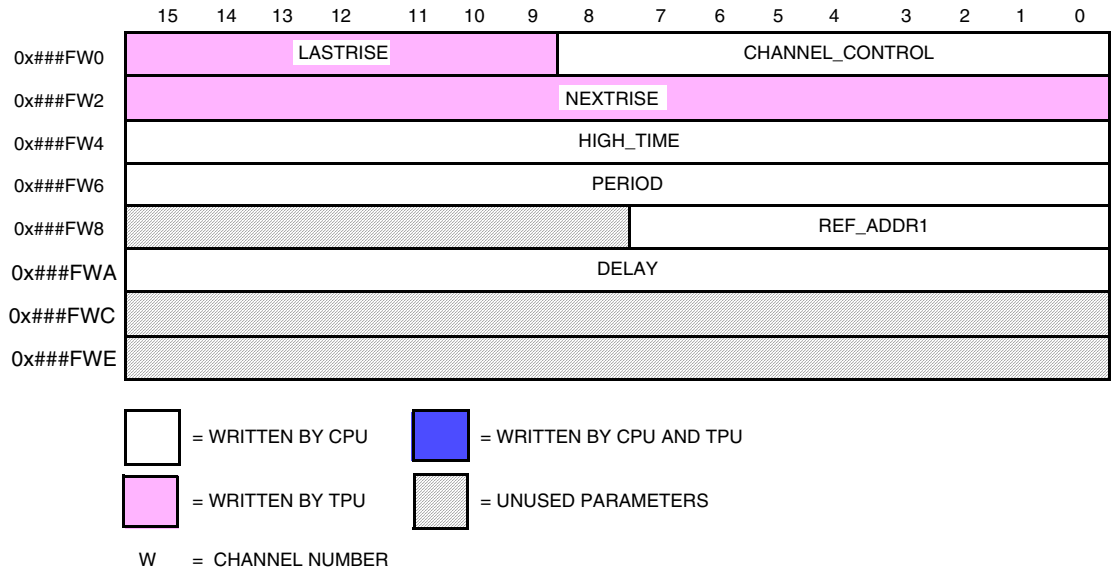
The SPWM output waveform duty cycle excludes 0% and 100%. If a PWM does not need to maintain a time relationship to another PWM, the PWM function should be used instead.

Figure D-25 and **Figure D-26** show all of the host interface areas for the SPWM function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES	
3 2 1 0	CHANNEL FUNCTION SELECT	SPWM FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	0x###E0C–0x###E12 0x###80C–0x###812	TPU3 A TPU3 B
1 0	CHANNEL PRIORITY	00 — DISABLE 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	0x###E14–0x###E16 0x###814–0x###816	TPU3 A TPU3 B
1 0	HOST SEQUENCE BITS	00 — MODE 0 01 — MODE 1 10 — MODE 2 11 — NOT USED	0x###E18–0x###E1A 0x###818–0x###81A	TPU3 A TPU3 B
1 0	HOST SERVICE BITS	00 — NO HOST SERVICE REQUEST 01 — NOT USED 10 — INITIALIZE 11 — IMMEDIATE UPDATE (MODE 1)	0x###E1C–0x###E1E 0x###81C–0x###81E	TPU3 A TPU3 B
0	INTERRUPT ENABLE	0 — INTERRUPT NOT ASSERTED 1 — INTERRUPT ASSERTED	0x###E0A 0x###80A	TPU3 A TPU3 B
0	INTERRUPT STATUS		0x###E20 0x###820	TPU3 A TPU3 B

PARAMETER RAM (MODE 0)



1030A-1

Figure D-25 SPWM Parameters, Part 1 of 2

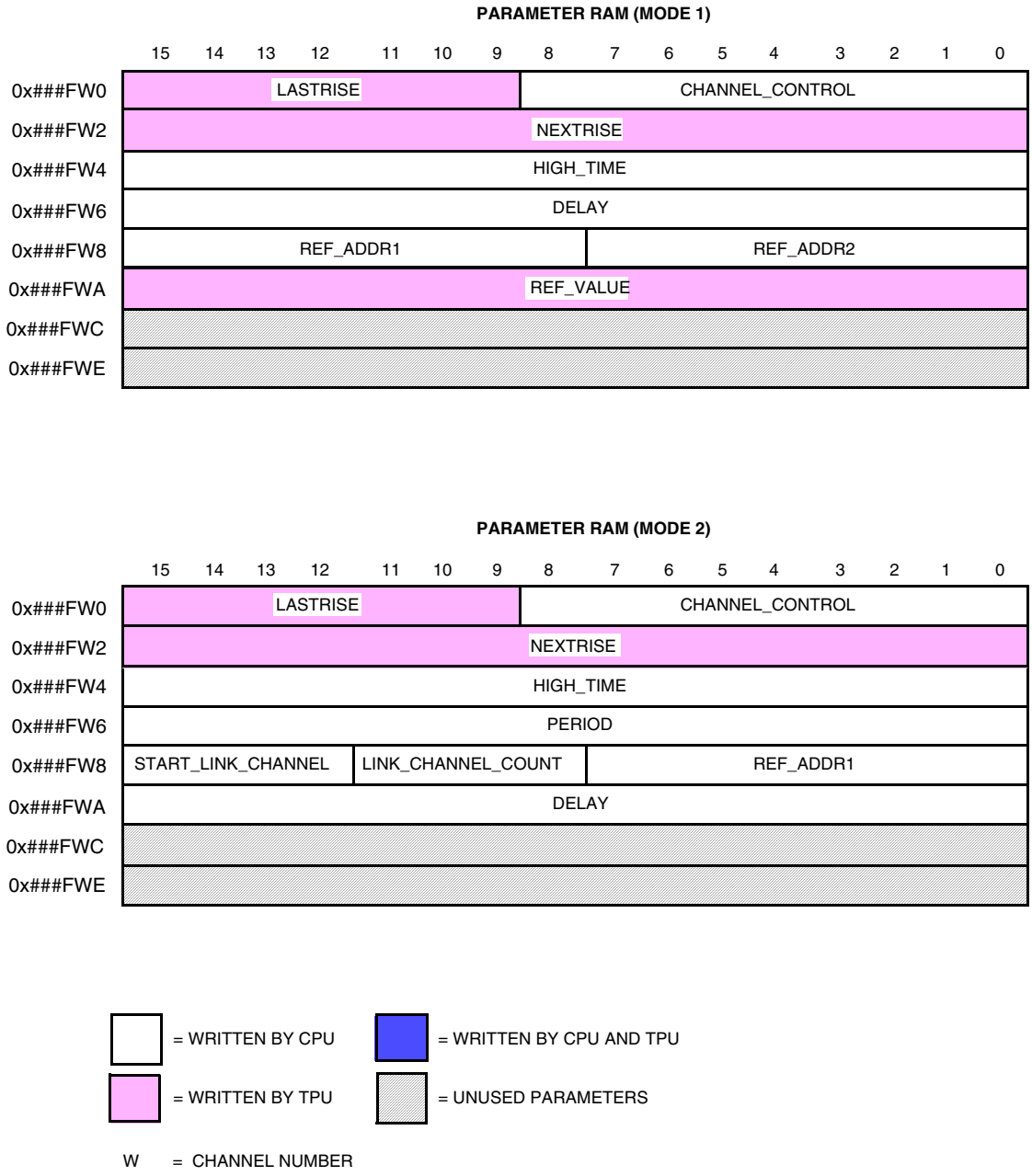


Figure D-26 SPWM Parameters, Part 2 of 2

D.17 Serial Input/Output Port (SIOP)

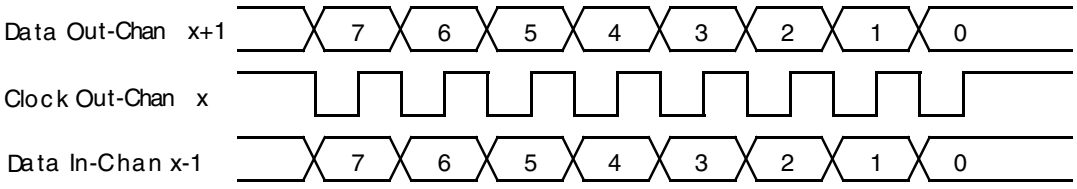
The serial input/output port (SIOP) TPU function uses two or three TPU channels to form a uni- or bi-directional synchronous serial port that can be used to communicate with a wide variety of devices. Features such as baud rate and transfer size are user programmable. The function can also produce a clock-only, when it uses just one channel.

The SIOP TPU function has been designed to closely resemble the SIOP hardware port found on some Motorola MCUs and can be used to add serial capabilities to a device without a serial port, or extend the capabilities of one with a hardware synchronous port.

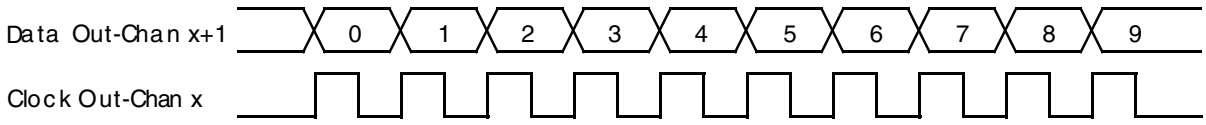
SIOP operates in master mode (i.e., the TPU always generates the clock) and the following features are programmable by the user:

1. Choice of clock-only (one channel), clock + transmit (two channels), clock + receive (two channels) or clock + transmit + receive (three channels) operating modes
2. Baud rate period is freely programmable by the user over a 15-bit range of TCR1 counts
3. Selection of msb or lsb first shift direction
4. Variable transfer size from one to 16 bits
5. Clock polarity is programmable

When a transfer of data is complete the SIOP function notifies the host CPU by issuing an interrupt request. The arrangement of the multiple SIOP channels is fixed: the data out channel is the channel above the clock channel and the data in channel is the channel below the clock channel. In clock-only or uni-directional mode, the unused TPU channels are free to run other TPU functions. Two possible SIOP configurations are show in **Figure D-27**.



8-bit bi-directional transfer, MSB first with data valid on clock rising edge



10-bit output only transfer, LSB first with data valid on clock falling edge

Figure D-27 Two Possible SIOP Configurations

D.17.1 Parameters

Figure D-28 shows the host interface areas and parameter RAM for the SIOP function. The following sections describe these parameters. Note that only the clock channel requires any programming by the user — the data in and out channels are entirely under TPU microcode control.

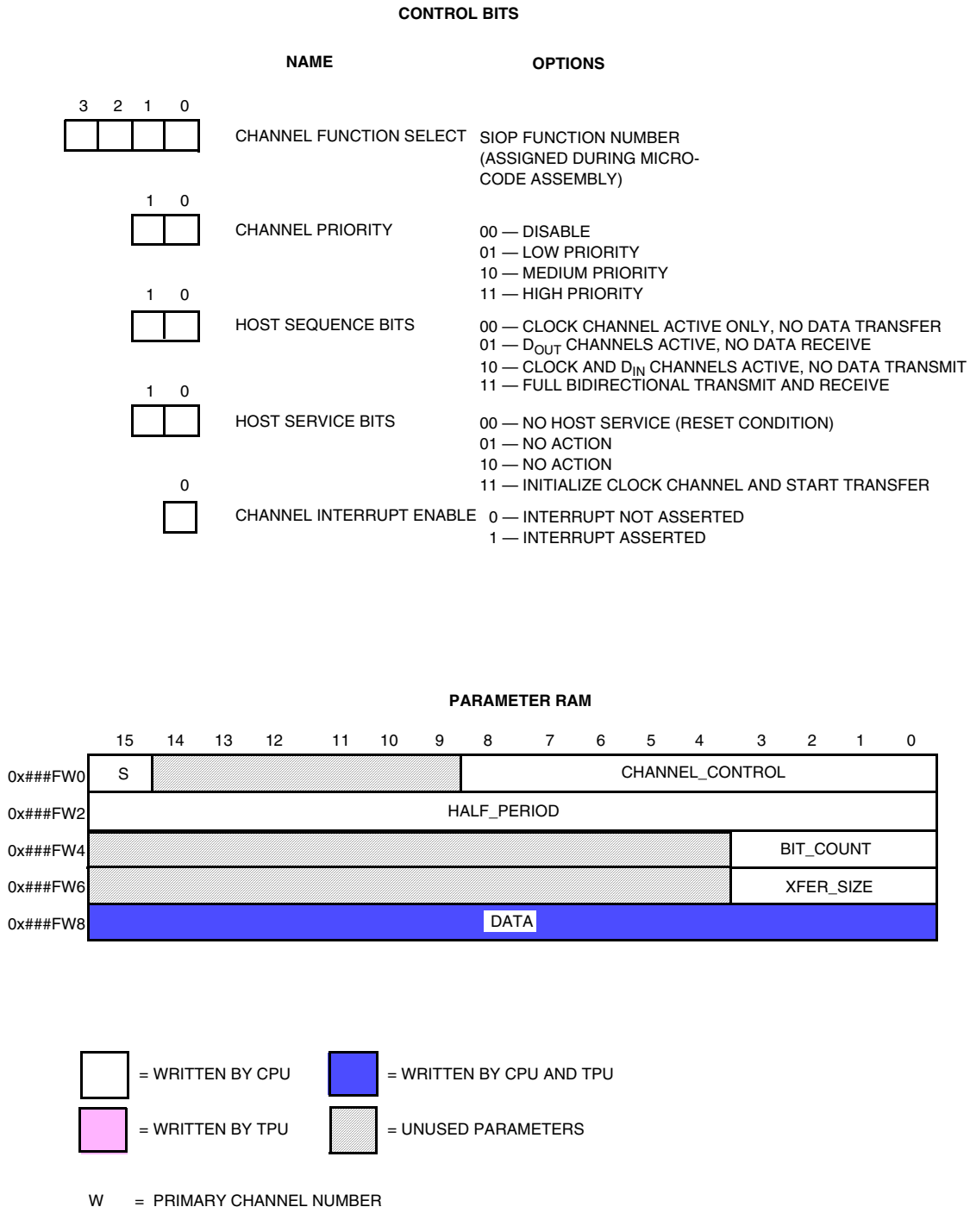


Figure D-28 SIOP Parameters

D.17.1.1 CHAN_CONTROL

This 9-bit CPU written parameter is used to setup the clock polarity for the SIOP data transfer. The valid values for CHAN_CONTROL for this function are given in the table below. CHAN_CONTROL must be written by the host prior to issuing the host service request (HSR) to initialize the function.

Table D-4 SIOP Function Valid CHAN_Control Options

CHAN_CONTROL[1] 8 7 6 5 4 3 2 1 0	Resulting Action
0 1 0 0 0 1 1 0 1	Data valid on clock falling edge.
0 1 0 0 0 1 1 1 0	Data valid on clock rising edge.

NOTES:
1. Other values of CHAN_CONTROL may result in indeterminate operation.

D.17.1.2 BIT_D

BIT_D is a CPU written bit that determines the direction of shift of the SIOP data. If BIT_D is zero then SIOP_DATA is right shifted (lsb first). If BIT_D is one then SIOP_DATA is left shifted (msb first).

D.17.1.3 HALF_PERIOD

This CPU-written parameter defines the baud rate of the SIOP function. The value contained in HALF_PERIOD is the number of TCR1 counts for a half SIOP clock period (e.g., for a 50 KHz baud rate, with a TCR1 period of 240 ns, the value [(1/50 KHz)/2]/240 ns = 42 should be written to HALF_PERIOD. The range for HALF_PERIOD is 1 to 0x8000, although the minimum value in practice will be limited by other system conditions. See notes on use and performance of SIOP function.

D.17.1.4 BIT_COUNT

This parameter is used by the TPU to count down the number bits remaining while a transfer is in progress. During the SIOP initialization state, BIT_COUNT is loaded with the value contained in XFER_SIZE. It is then decremented as the data is transferred and when it reaches zero, the transfer is complete and the TPU issues an interrupt request to the CPU.

D.17.1.5 XFER_SIZE

This CPU-written parameter determines the number of bits that make up a data transfer. During initialization, XFER_SIZE is copied into BIT_COUNT. XFER_SIZE is shown as a 5-bit parameter to match the maximum size of 16 bits in SIOP_DATA, although the TPU uses the whole word location. For normal use, XFER_SIZE should be in the range 1-to-16.

D.17.1.6 SIOP_DATA

This parameter is the data register for all SIOP transfers. Data is shifted out of one end of SIOP_DATA and shifted in at the other end, the shift direction being determined by the value of BIT_D. In output only mode, zero will be shifted into SIOP_DATA and

in input only mode, the data shifted out is ignored. In clock-only mode SIOP_DATA is still shifted. Note that no 'justifying' of SIOP_DATA is performed by the TPU, (e.g., if an 8-bit bi-directional transfer is made, shifting lsb first, then the bottom byte of SIOP_DATA will be shifted out and the input data will be shifted into the upper byte of SIOP_DATA).

NOTE

SIOP_DATA is not buffered. The CPU should only access it between completion of one transfer and the start of the next.

D.17.2 Host CPU Initialization of the SIOP Function

The CPU initializes the SIOP function by:

1. Disabling the channel by clearing the two channel priority bits
2. Selecting the SIOP function on the channel by writing the assigned SIOP function number to the function select bits
3. Writing CHAN_CONTROL in the clock channel parameter RAM
4. Writing HALF_PERIOD, BIT_D and XFER_SIZE in the clock channel parameter RAM to determine the speed, shift direction and size of the transfer
5. Writing SIOP_DATA if the data output is to be used
6. Selecting the required operating mode via the two host sequence bits
7. Issuing a host service request type %11
8. Enabling service by assigning H, M or L priority to the clock channel via the two channel priority bits

The TPU then starts the data transfer, and issues an interrupt request when the transfer is complete.

Once the function has been initialized, the CPU only needs to write SIOP_DATA with the new data and issue a HSR %11 to initiate a new transfer. In input or clock-only modes, just the HSR %11 is required.

D.17.3 SIOP Function Performance

Like all TPU functions, the performance limit of the SIOP function in a given application is dependent to some extent on the service time (latency) associated with other active TPU channels. This is due to the operational nature of the scheduler. Where two channels are being used for a uni-directional system, and no other TPU channels are active, the maximum baud rate is approximately 230 KHz at a bus speed of 16.77 MHz. A three-channel bi-directional system under the same conditions has a maximum baud rate of approximately 200 KHz. When more TPU channels are active, these performance figures will be degraded, however, the scheduler assures that the worst case latency in any TPU application can be closely approximated. It is recommended that the guidelines given in the TPU reference manual be used along with the information given in the SIOP state timing table to perform an analysis on any proposed TPU application that appears to approach the performance limits of the TPU.

Table D-5 SIOP State Timing[1]

State Number and Name	Max. CPU Clock Cycles	Number of RAM Accesses by TPU
S1 SIOP_INIT		
HSQ = X0	28	7
X1	38	7
S2 DATA_OUT		
HSQ = X0	14	4
X1	24	4
S3 DATA_IN		
HSQ = 0X	14	4
1X	28	6

NOTES:
1. Execution times do not include the time slot transition time (TST = 10 or 14 CPU clocks).

D.17.3.1 XFER_SIZE Greater than 16

XFER_SIZE is normally programmed to be in the range 1-to-16 to match the size of SIOP_DATA, and has thus been shown as a 5-bit value in the host interface diagram. However, the TPU actually uses all 16 bits of the XFER_SIZE parameter when loading BIT_COUNT. In some unusual circumstances this can be used to the user's advantage. If an input device is producing a data stream of greater than 16 bits then manipulation of XFER_SIZE will allow selective capturing of the data. In clock-only mode, the extended XFER_SIZE can be used to generate up to 0xFFFF clocks.

D.17.3.2 Data Positioning

As stated above, no 'justifying' of the data position in SIOP_DATA is performed by the TPU. This means that in the case of a byte transfer, the data output will be sourced from one byte and the data input will shift into the other byte. This rule holds for all data size options except 16 bits when the full SIOP_DATA register is used for both data output and input.

D.17.3.3 Data Timing

In the example given in **Figure D-29**, the data output transitions are shown as being completely synchronous with the relevant clock edge and it is assumed that the data input is latched exactly on the opposite clock edge. This is the simplest way to show the examples, but is not strictly true. Since the TPU is a multi-tasking system, and the data channels are manipulated directly by microcode software while servicing the clock edge, there is a finite delay between the relevant clock edge and the data-out being valid or the data-in being latched. This delay is equivalent to the latency in servicing the clock channel due to other TPU activity and is shown as 'Td' in the timing diagram. Td is the delay between the clock edge and the next output data being valid and also the delay between the opposite clock edge and the input data being read. For the vast majority of applications, the delay Td will not present a problem and can be ignored. Only for a system which heavily loads the TPU should the user calculate the worst case latency for the SIOP clock channel + actual SIOP service time (= Td) and ensure that the baud rate is chosen such that HALF_PERIOD - Td is not less that the

minimum setup time of the receiving device. A transmitting device must also hold data valid for a minimum time of Td after the clock.

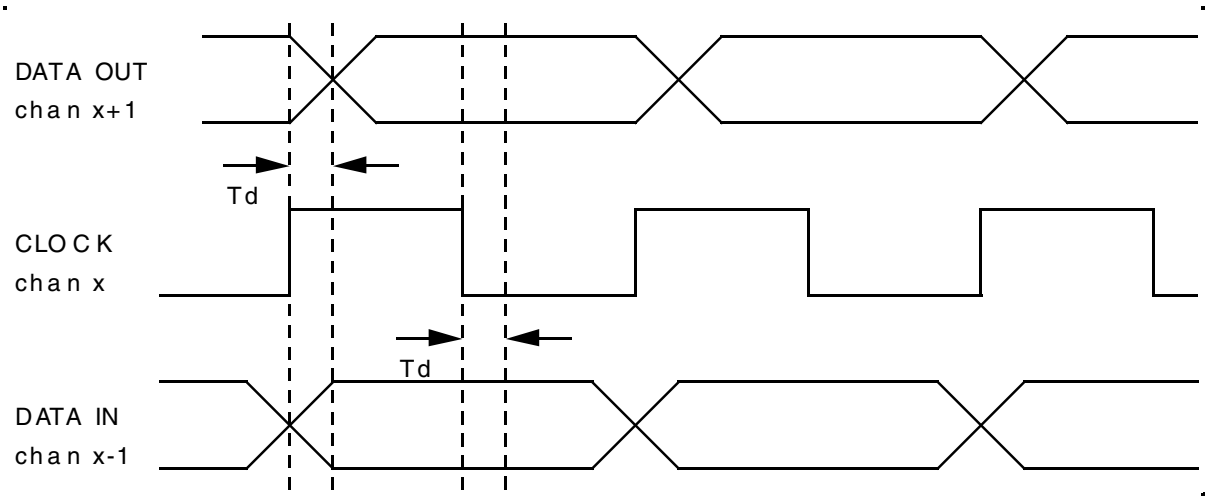


Figure D-29 SIOP Function Data Transition Example

APPENDIX E
ELECTRICAL AND AC CHARACTERISTICS

E.1 Absolute Maximum Ratings

Table E-1 Maximum Ratings (V_{SS} = 0 V)

Rating	Symbol	Value	Unit
3.3-V Supply Voltage	V_{DDI}, V_{DD3}, $V_{DDDPTRAM}$, V_{STBY}	-0.3 to + 4.0	V
Clock Sythesizer Voltage	V_{DDSYN}	-0.3 to 4.0	V
QADC Supply Voltage	V_{DDA}	-0.3 to 6.0	V
5-V Supply Voltage	V_{DD5}	-0.3 to 6.0	V
DC Input Voltage[10]	V_{IN}	-0.3 to 5.5	V
Operating Temperature Range (Packaged)	T_A	-40 to +125	°C
Operating Temperature Range (Die Form)	T_J	-40 to +150	°C
Storage Temperature Range	T_{STG}	–55 to 150	°C
Maximum Input Current per Pin[7,8]	I_{MAX}	1.0	mA

NOTES:

1. Permanent damage can occur if maximum ratings are exceeded. Exposure to voltages or currents in excess of recommended values affects device reliability. Device modules may not operate normally while being exposed to electrical extremes.
2. Although sections of the device contain circuitry to protect against damage from high static voltages or electrical fields, take normal precautions to avoid exposure to voltages higher than maximum-rated voltages.
4. All functional non-supply pins are internally clamped to V_{SS}. All functional pins except XTAL and EXTAL are internally clamped to V_{DD}.
5. Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values for positive and negative clamp voltages, then use the larger of the two values.
6. Power supply must maintain regulation within operating V_{DD} range during instantaneous and operating maximum current conditions.
7. This parameter is periodically sampled rather than 100% tested
8. Total input current for all digital input-only and all digital input/output pins must not exceed 10 mA. Exceeding this limit can cause disruption of normal operation.
10. All 3-V input pins are 5-V tolerant except XTAL and EXTAL.

Table E-2 Thermal Characteristics

Rating	Symbol	Value	Unit
Thermal Resistance Plastic 272-Pin Ball Grid Array	Θ_{JA}	TBD	°C/W

NOTES:

The average chip-junction temperature (T_J) in °C can be obtained from:

$$T_J = T_A + (P_D \times \Theta_{JA}) \quad (1)$$

where

T_A = Ambient Temperature, °C

Θ_{JA} = Package Thermal Resistance, Junction-to-Ambient, °C/W

$P_D = P_{INT} + P_{I/O}$

$P_{INT} = I_{DD} \times V_{DD}$, Watts — Chip Internal Power

$P_{I/O}$ = Power Dissipation on Input and Output Pins — User Determined

An approximate relationship between P_D and T_J (if $P_{I/O}$ is neglected) is:

$$P_D = K \div (T_J + 273\ °C) \quad (2)$$

Solving equations 1 and 2 for K gives:

$$K = P_D + (T_A + 273°C) + \Theta_{JA} \times P_D^2 \quad (3)$$

where K is a constant pertaining to the particular part. K can be determined from equation (3) by measuring P_D (at equilibrium) for a known T_A. Using this value of K, the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.

Table E-3 General DC Electrical Characteristics

Characteristic	Symbol	Min	Max	Unit
3V Input High Voltage [1] Except EXTAL and XTAL	V_{IH3}	2.0	V_{DD5}	V
5v Input High Voltage[2] All input pins except TPU and CTM9 TPU and CTM9 input pins	V_{IH5}	0.7 (V_{DD5}) 3.3	V_{DD5} + 0.3 V_{DD5} + 0.3	V V
3-V Input Low Voltage	V_{IL3}	V_{SS} - 0.3	0.8	V
5-V Input Low Voltage All input pins except TPU and CTM9 TPU and CTM9 input pins	V_{IL5}	V_{SS} – 0.3 V_{SS} – 0.3	0.35 (V_{DD5}) 2.3	V V
Input Hysteresis[11] All pins except TPU and CTM9 TPU and CTM9 input pins	V_{HYS}	0.5 0.3	-	V V
Mode Select Pullup/Pulldown Current	I_{ACT}	20	130	µA
Input Leakage Current[3] Pull-up/down inactive	I_{INACT}	-1.0	1.0	µA
3-V Output High Voltage (I_{OH3} = -2 mA)	V_{OH3}	2.4	-	V
5-V Output High Voltage (I_{OH5} = -2 mA)	V_{OH5}	V_{DD5} - 0.7	-	V
3-V Output Low Voltage (I_{OL3} = 3.2 mA)	V_{OL3}	-	0.5	V
5-V Output Low Voltage (I_{OL5} = 2 mA)	V_{OL5}	-	0.5	V
Clock Output Low Current CLKOUT or BCLK @ V_{OL3} = 0.5 V	I_{OL}	2.0	-	mA

Table E-3 General DC Electrical Characteristics (Continued)

Characteristic	Symbol	Min	Max	Unit
Load for BIM 3-V Bus Output Pins, 25 MHZ[9] Partial Drive Full Drive	C_{LBIM25}	-	 40 80	pF
Load for BIM 3-V Bus Output Pins, 33 MHZ[9] Partial Drive Full Drive	C_{LBIM33}	-	 25 50	pF
Load for Visibility Bus 3-V Output Pins[9] Load for QSM 5V Output Pins[9] Primary Functions Digital Output	C_{LVB} C_{LQSM}	-	25 200 50	pF
Capacitance for Input, Output, and Bidirectional[9] V_{in} = 0V, f = 1MHz (except QADC)	C_{IN}	-	15	pF
Supply Current 25 MHZ[6,11,12] RUN LPSTOP, 10 MHz crystal, VCO Off LPSTOP (External clock input frequency = max f_{sys})	 I_{DD3} I_{DD5} I_{DDXTAL} $I_{DDEXTCK}$		 215 20 **TBD** **TBD**	mA
Supply Current 33 MHZ[6,11,12] RUN LPSTOP, 10 MHz crystal, VCO Off LPSTOP (External clock input frequency = max f_{sys})	 I_{DD3} I_{DD5} I_{DDXTAL} $I_{DDEXTCK}$		 **TBD** **TBD** **TBD** **TBD**	mA
PLL Supply Current (V_{DDSYN})[6,12] 10 MHz crystal, VCO on, maximum f_{sys} External Clock, maximum f_{sys} LPSTOP, 10 MHz crystal, VCO off (STBIM = 0) LPSTOP, 10 MHz crystal, VCO on (STBIM = 1) LPSTOP, 10 MHz crystal, VCO off (STCLKS = 1)	 I_{DDSYN} EXI_{DDSYN} SI_{DDSYNB} SI_{DDSYNV} $SI_{DDSYNST}$		 10.0 **TBD** **TBD** **TBD** **TBD**	 mA mA mA mA µA
FASRAM Standby Voltage (VSTBY)[7] Run Standby mode, $V_{DD} = V_{SS}$	V_{SB}	3.14 2.5	3.45 3.45	V
FASRAM Standby Current [6,7,10,11] Normal RAM operation$V_{DD} > V_{SB} - 0.5$ V Transient condition$V_{SB} - 0.5$ V >=V_{DD} >=V_{SS} + 0.5 V Standby operation$V_{DD} < V_{SS} + 0.5$ V	 I_{SB}	 **TBD**	 **TBD** **TBD** **TBD**	 µA mA µA
DC Injection Current per Pin, 3V/5V[3,9]	I_{IC}	-1.0	1.0	mA

NOTES:

1. This spec is for 3V outputs with 5-V tolerent input pins.
2. This spec is for 5V outputs with 5-V tolerent input pins.
3 After characterization this value may be improved.
6. Power dissipation measured at 32 Mhz or 25 Mhz system clock frequency, all modules active. Power dissipation can be calculated using the following expression: P_D = Maximum V_{DD5} (I_{DD5} + I_{DDA}) + Maximum V_{DD3} (I_{DD3} + I_{DDSYN} + I_{SB})
7. The RAM module will not switch into standby mode as long as V_{SB} does not exceed V_{DD} by more than 0.5 Volt. The RAM array cannot be accessed while the module is in standby mode. V_{SB} must be applied to the RAM during normal operation.

8. Power dissipation measured at 32 Mhz or 25 Mhz system clock frequency, all modules active.
9. This parameter is periodically sampled rather than 100% tested.
10. When V_{DD} is transitioning during power-up or power down sequence, and V_{SB} is applied, current flows between the V_{STBY} and V_{DD} pins, which causes standby current to increase toward the maximum transient condition specification. System noise on the V_{DD} and V_{STBY} pins can contribute to this condition.
11. These values are design targets, Motorola makes no commitment to achieve this level of electrical performance.
12. I_{DD3} and I_{DD5} are estimates based on average current calculations.

E.2 AC Specifications

Each timing diagram contains parameters abstracted from the timing specification tables. Pertinent notes have been included in the timing tables.

In some cases a timing parameter has been augmented based on different signal options. The user must select the appropriate parameter.

EXAMPLE

Timing parameter 9 has been augmented to include specific timing for: (9) Clock low to CS Valid (CS set to DS timing), (9A) Clock low to AS/DS Valid, and (9B) Clock low to CS Valid (CS set to CE timing).

AC timing is measured under the following conditions:

- Timing is measured between 20% V_{DD} and 70% V_{DD}.
- Temperature: T_A in normal operating range.
- Voltage: V_{DD3}, V_{DDi}, V_{STBY}, V_{DDSYN} = 3.3 V ± 5%; V_{DD5}, V_{DDA} = 5.0 V ± 5%.

Characterization of device emissions are performed per *SAE J1752/3* Issued March 1995.

E.2.1 PLL AC Timing

Table E-4 PLL Timing

(10 MHz Reference)

Num	Characteristic	Symbol	Min	Max	Unit
C1	PLL Reference Frequency Range Crystal reference External reference[10]	 $f_{ref\_crystal}$ f_{ref_ext}	 2 10	 10 32	MHz
C2	System Frequency On-Chip PLL Frequency	f_{sys}	dc 0.032	32 32	MHz
C3	Loss of Reference Frequency[2,8]	f_{lor}	1.5	3.0	% fsys
C4	Self Clocked Mode Clock Frequency[2,9]	f_{scm}	.5	32	MHz
C5	Crystal Start-up Time[2,4]	t_{cst}		20	ms
C6	PLL Lock Time[2,3,4,5]	t_{lpll}		200	µs
C7	Power-To-Lock Time[2,3,4,6] With Crystal Reference Without Crystal Reference	t_{lplk}		 21 200	 ms µs
C8	CLKOUT Period Jitter[2,3,4,5,7] Peak-to-peak Jitter Short Term Jitter (3 system clock cycles interval) Long Term Jitter (2 msec interval)	Cjitter		 3 1 .01	 % % %
C9	Duty Cycle of reference[2]	tdc	40	60	%
C10	1:1 Clock Skew (between CLKOUT and EXTAL)[10]	tskew	-2	2	ns

NOTES:

2. This parameter is periodically sampled rather than 100% tested.
3. Presently, the filter capacitor is implemented on chip, but any future application without an internal filter assumes that low-leakage external filter capacitors are attached to the XFCN and XFCP pins. Total external resistance from either of the XFCN or XFCP pins due to external leakage sources to either V_{DD} or V_{SS} must be greater than 10 MΩ to guarantee this specification.
4. Proper layout procedures must be followed to achieve specifications.
5. This specification applies to the period required for the PLL to relock after changing the MFD frequency control bits in the synthesizer control register (SYNCR), and to the period required for the PLL to relock on exiting LP-STOP.
6. Assuming a reference is available at power up, lock time is measured from the time V_{DD} and V_{DDSYN} are valid to $\overline{\text{RESET}}$ release. If the crystal oscillator is being used as the reference for the PLL, then the crystal start up time must be added to the PLL lock time to determine the total start-up time.
7. Jitter is the average deviation from the programmed frequency measured over the specified interval at maximum f_{sys}. Measurements are made with the device powered by filtered supplies and clocked by a stable external clock signal. Noise injected into the PLL circuitry via V_{DDSYN}, V_{SS}, and variations in crystal oscillator frequency increase the C_{jitter} percentage for a given interval.
8. Loss of reference frequency is the reference frequency detected internally which transitions the PLL into self clocked mode. Loss of reference frequency is specified as a percentage of the operating frequency, fsys.
9. Self clocked mode frequency is the frequency that the PLL operates at when the reference frequency falls below f_{lor} with default MFD/RFD settings.
10. PLL is operating in 1:1 PLL mode.

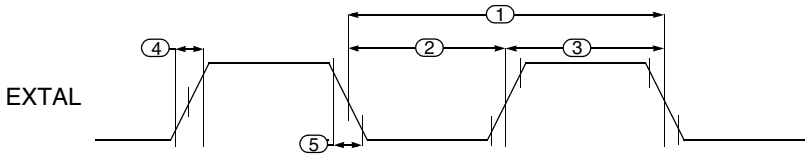


Figure E-1 External / 1:1 PLL CLOCK Input Timing

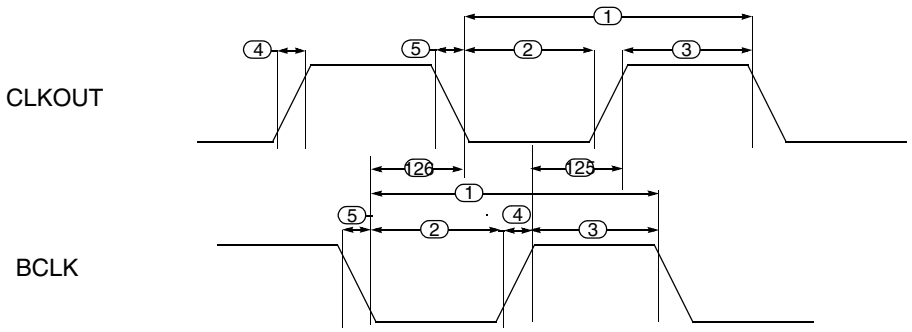


Figure E-2 CLKOUT and BCLK Output Timing

E.2.2 BIM AC Timing

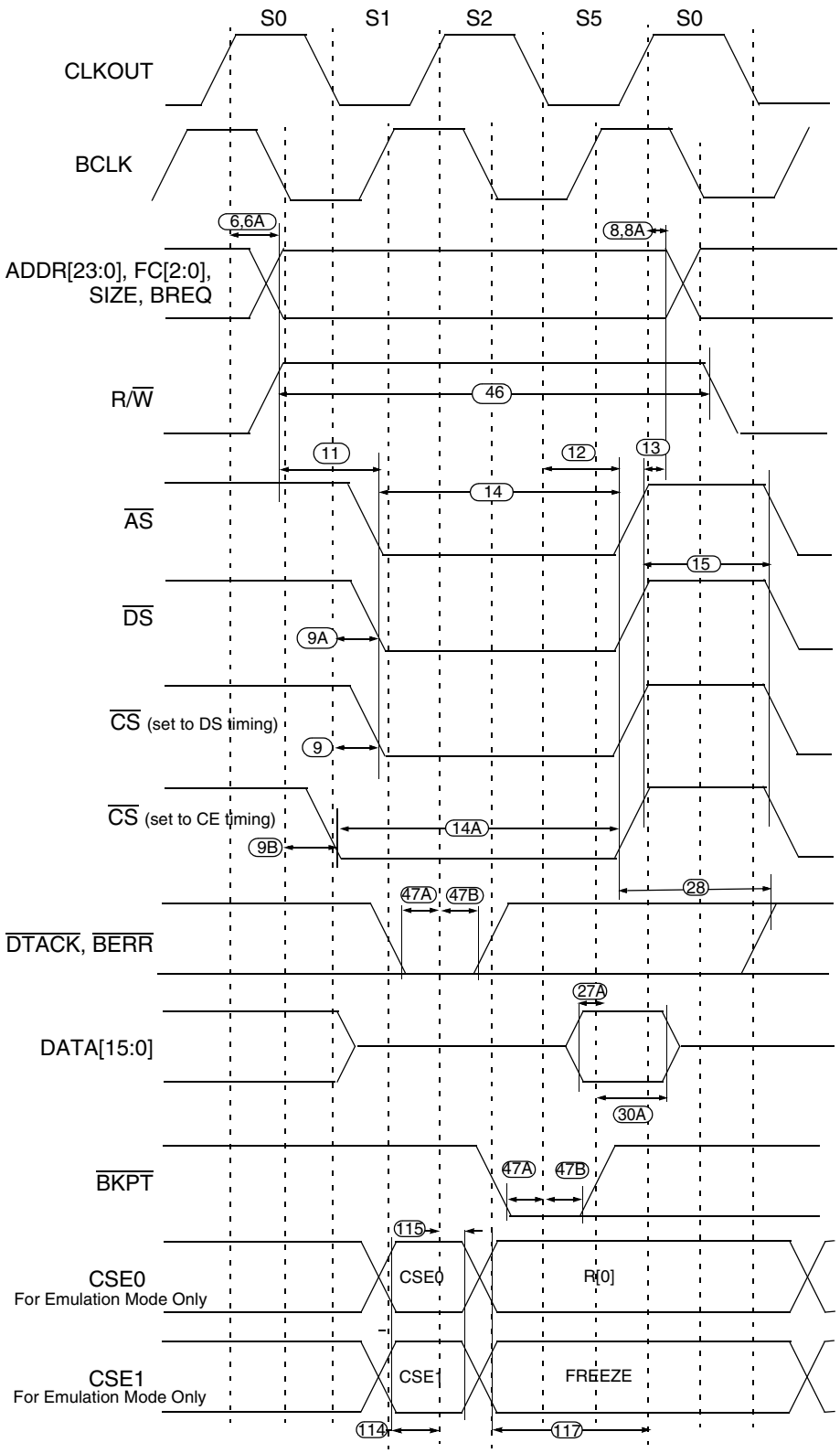
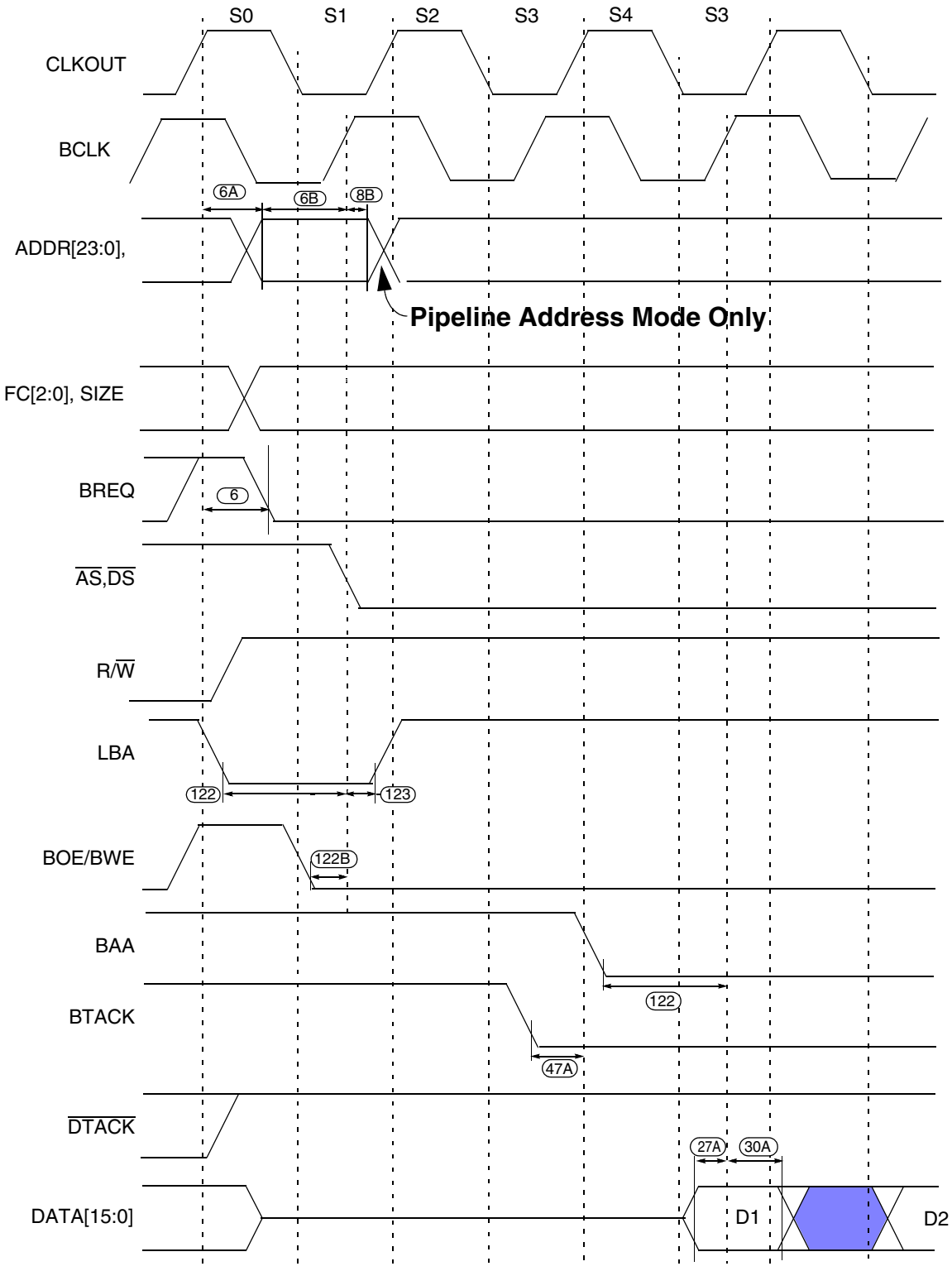


Figure E-3 Read Cycle



Note: Bus cycle is shown with HPCE asserted - LBA asserts in S0.

Figure E-4 Burst Read Cycle — Initiation

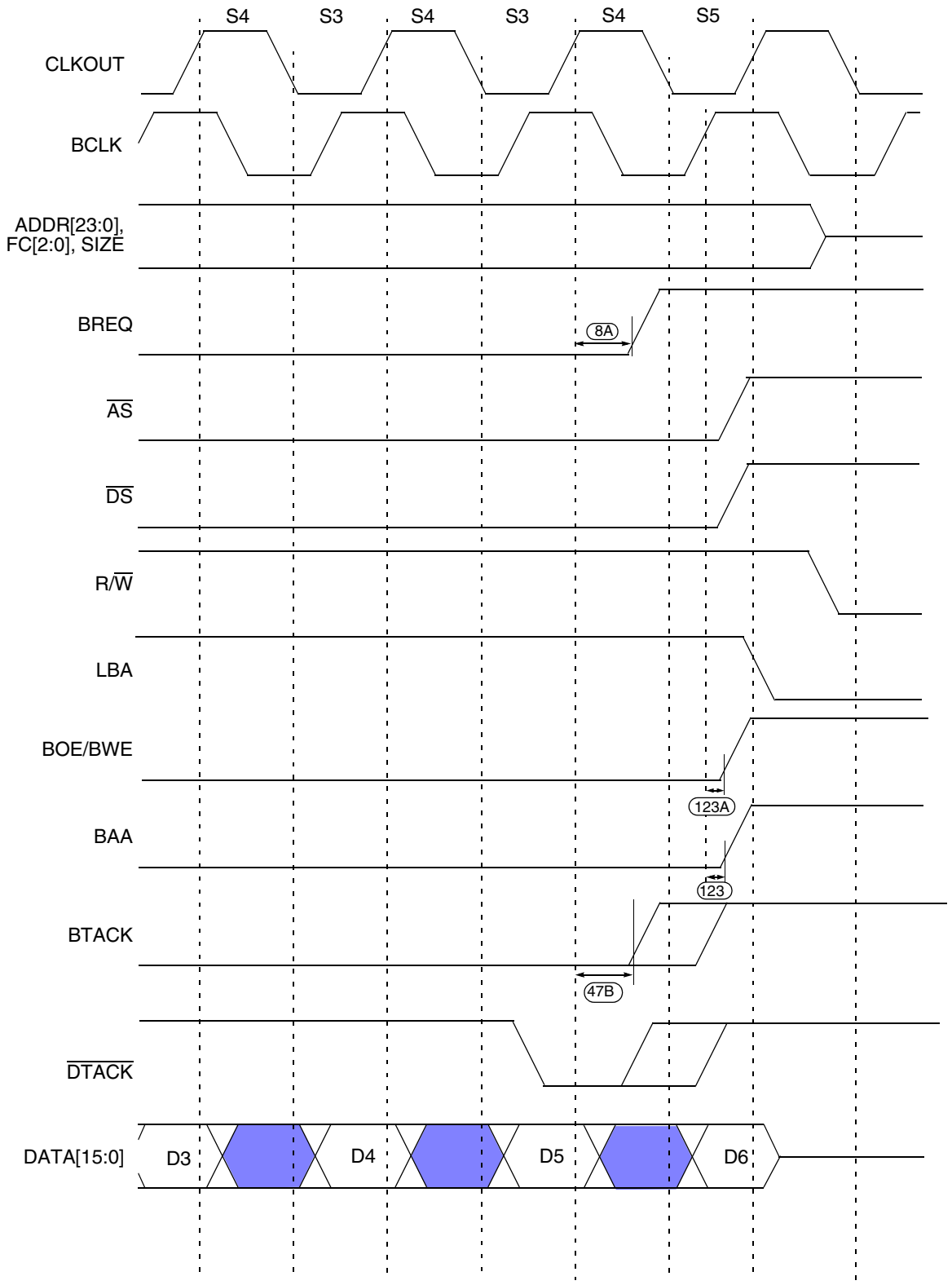


Figure E-5 Burst Read Cycle — Completion

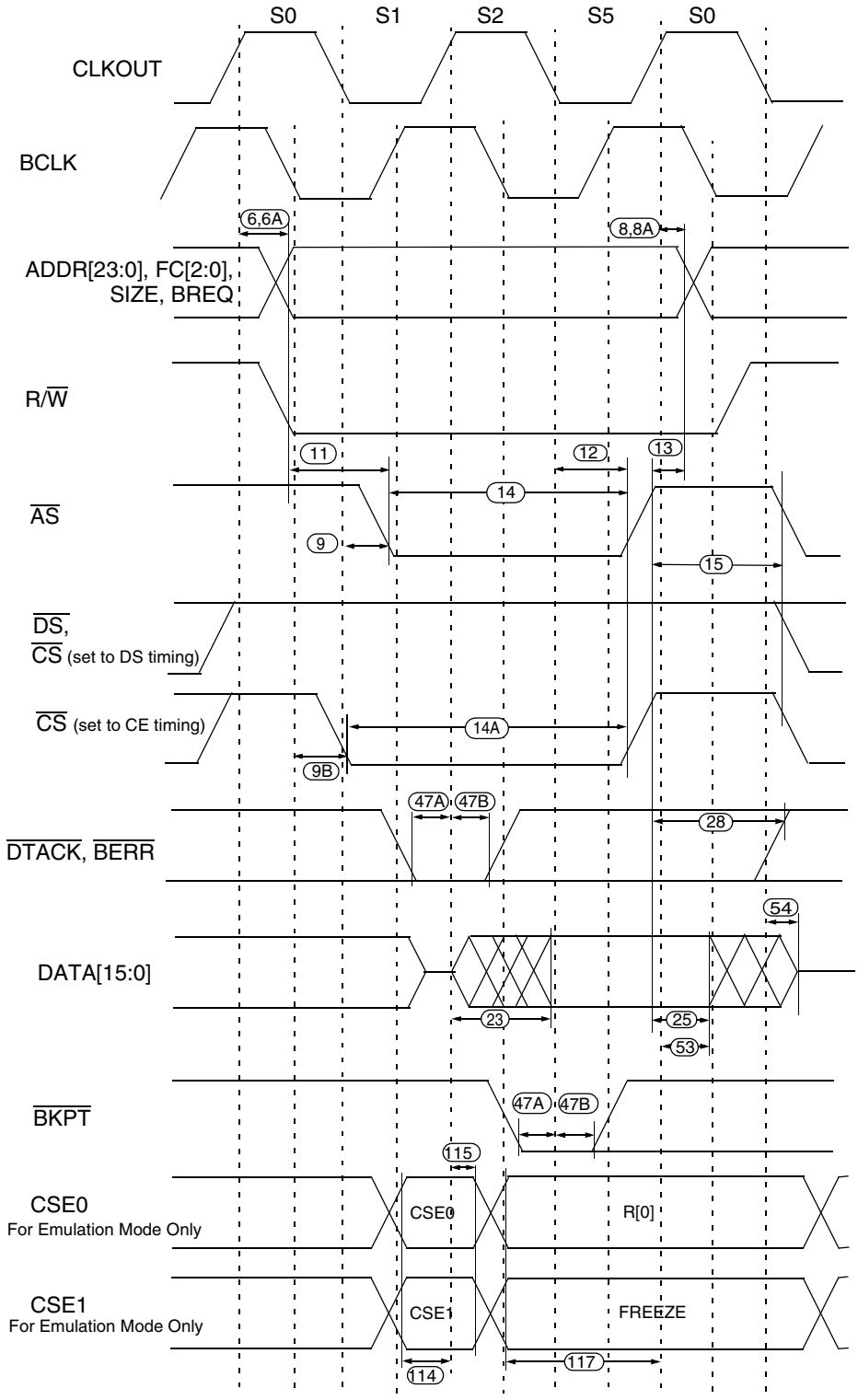


Figure E-6 Write Cycle

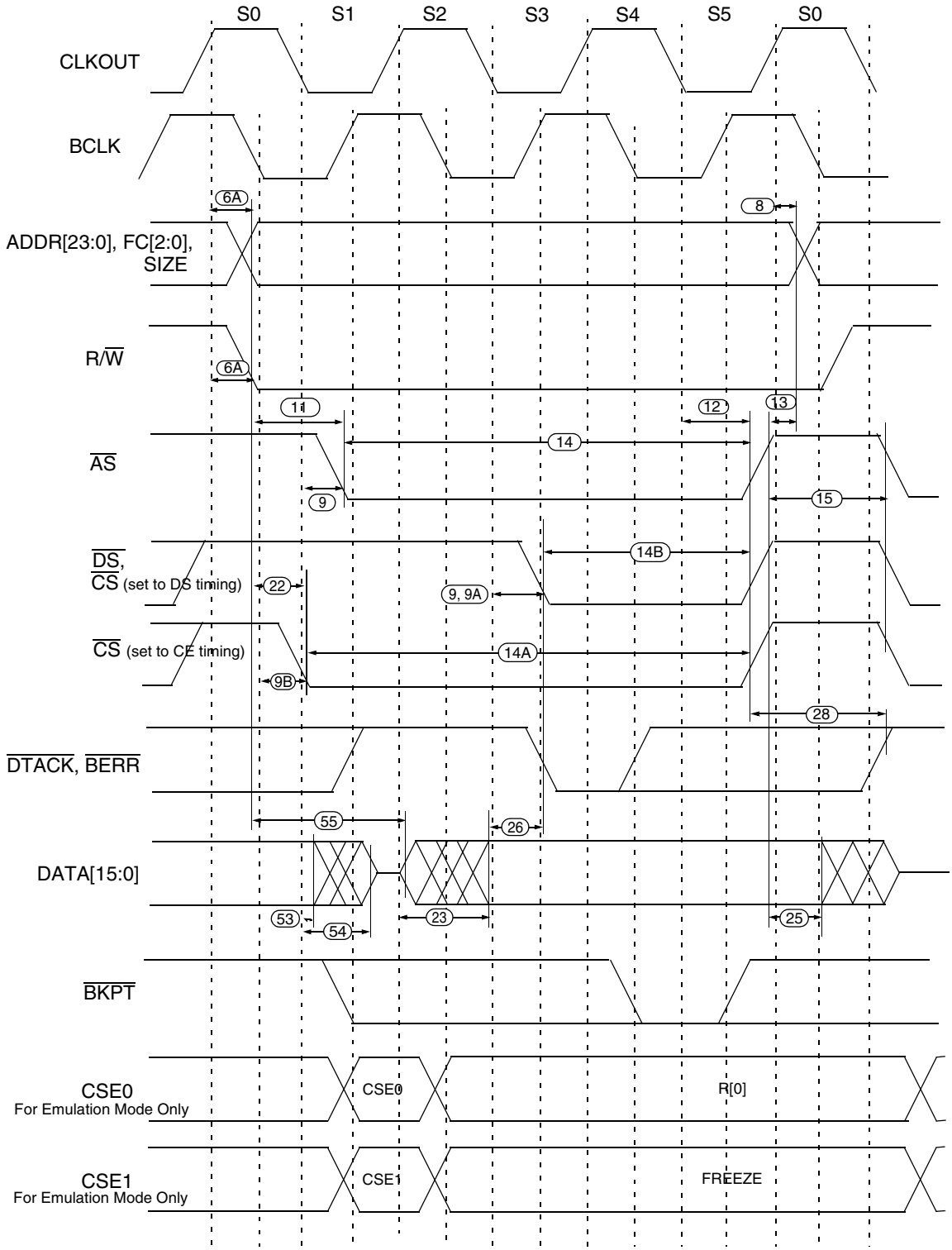
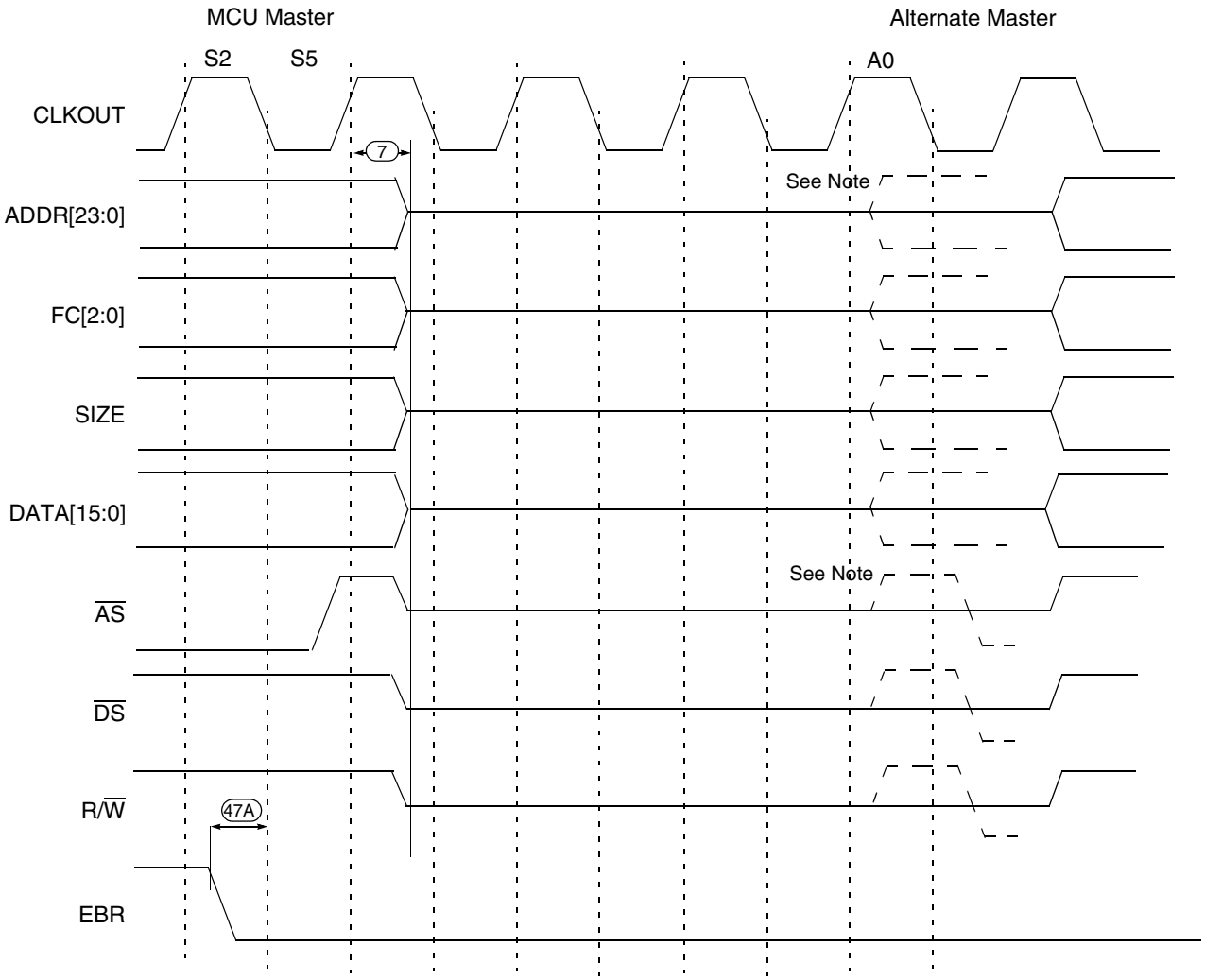


Figure E-7 Write Cycle With One Wait State



Note: The EBI may run one or more bus cycles after EBR is asserted. An external master may not start a bus cycle until the AS and DS pins have been negated for three clocks.

Figure E-8 Alternate Bus Mastership

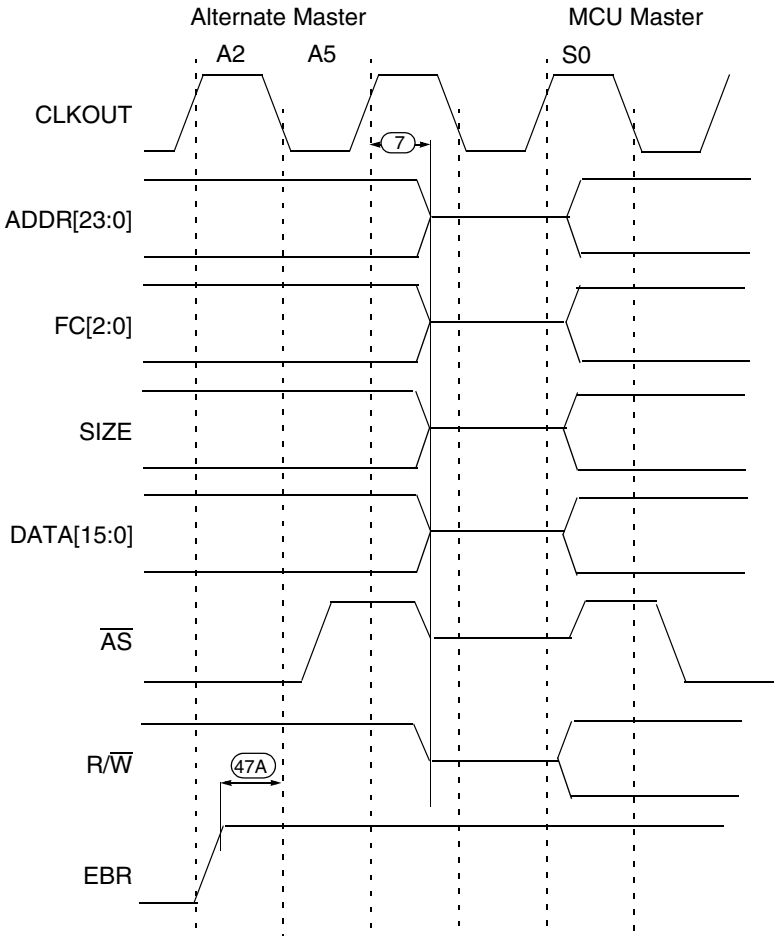
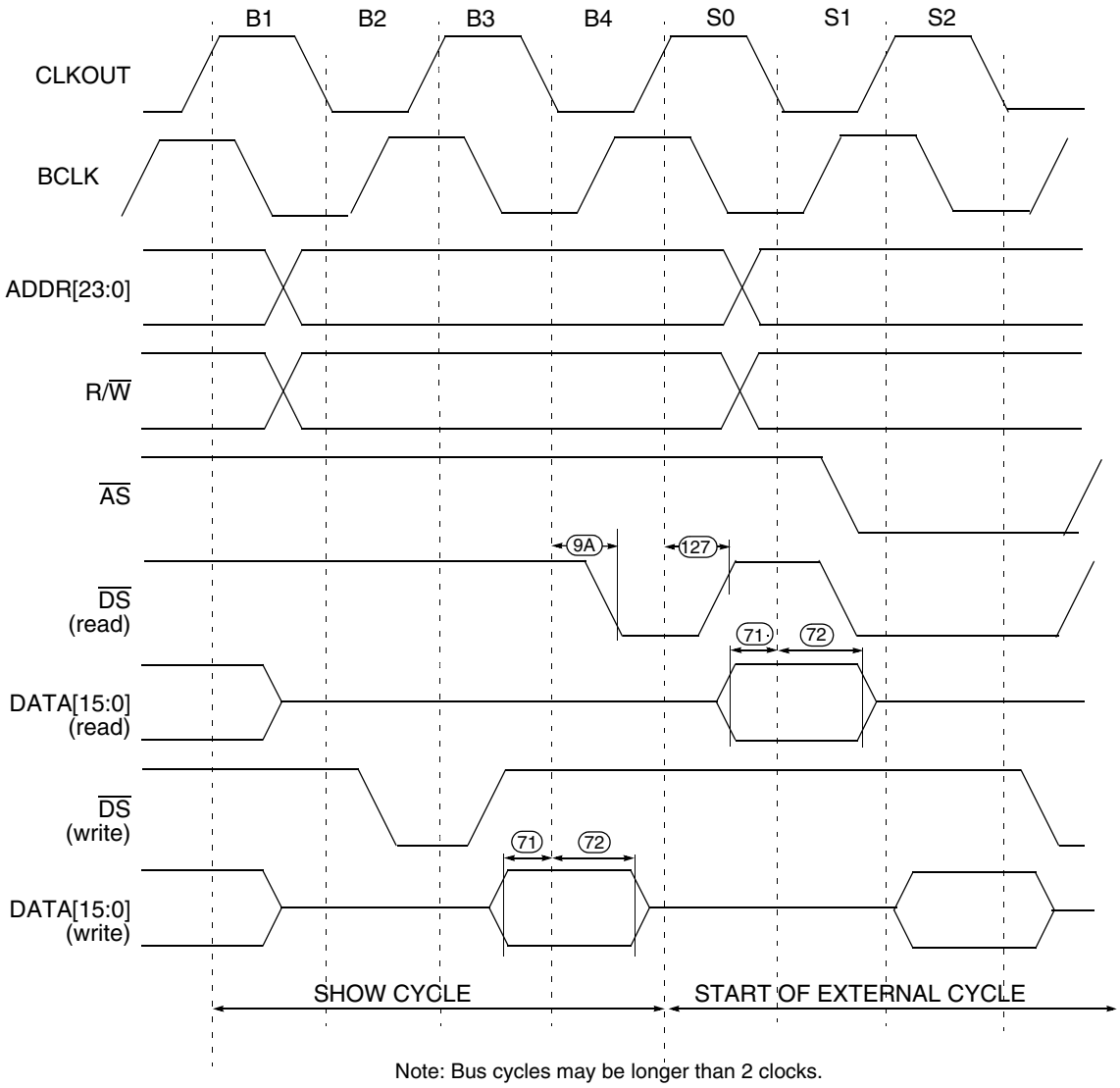


Figure E-9 EBR — Alternate Master Bus Release



Note: Bus cycles may be longer than 2 clocks.

Figure E-10 Non-Burst Show Cycle Timing

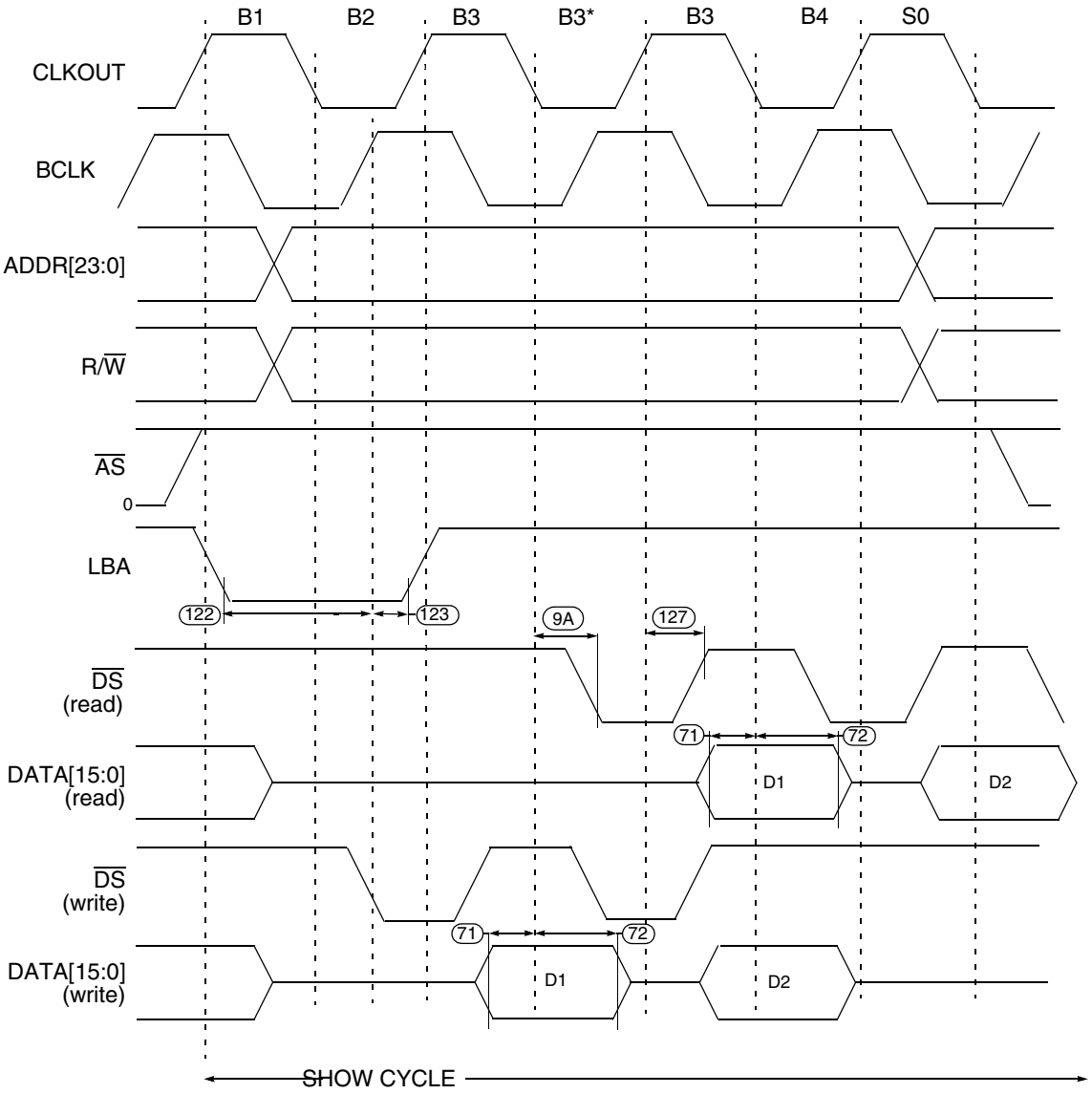


Figure E-11 Burst Show Cycle Timing

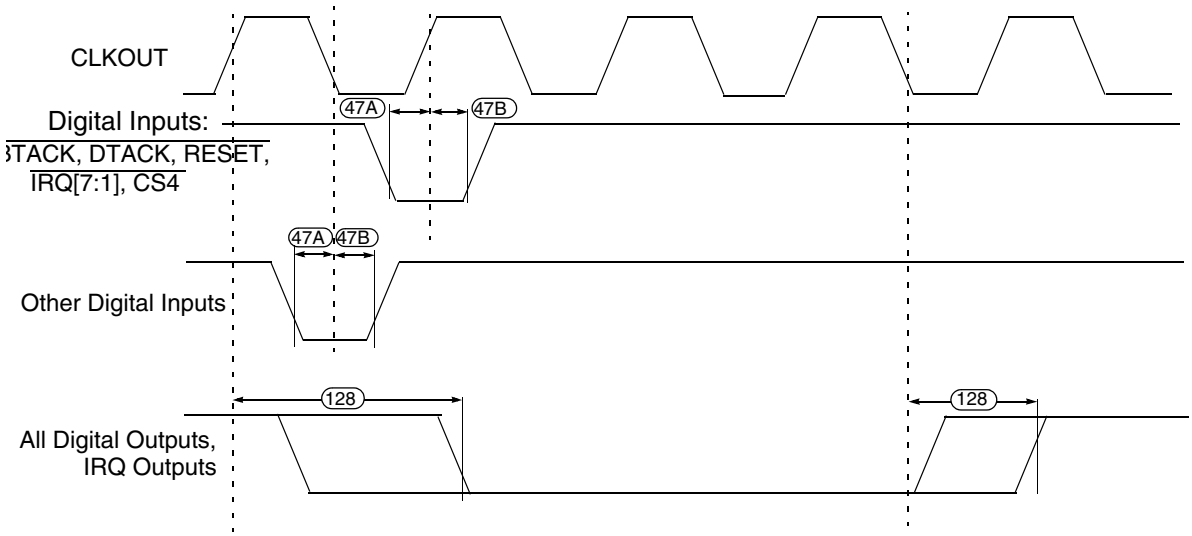


Figure E-12 Digital Input/Output and IRQ Output

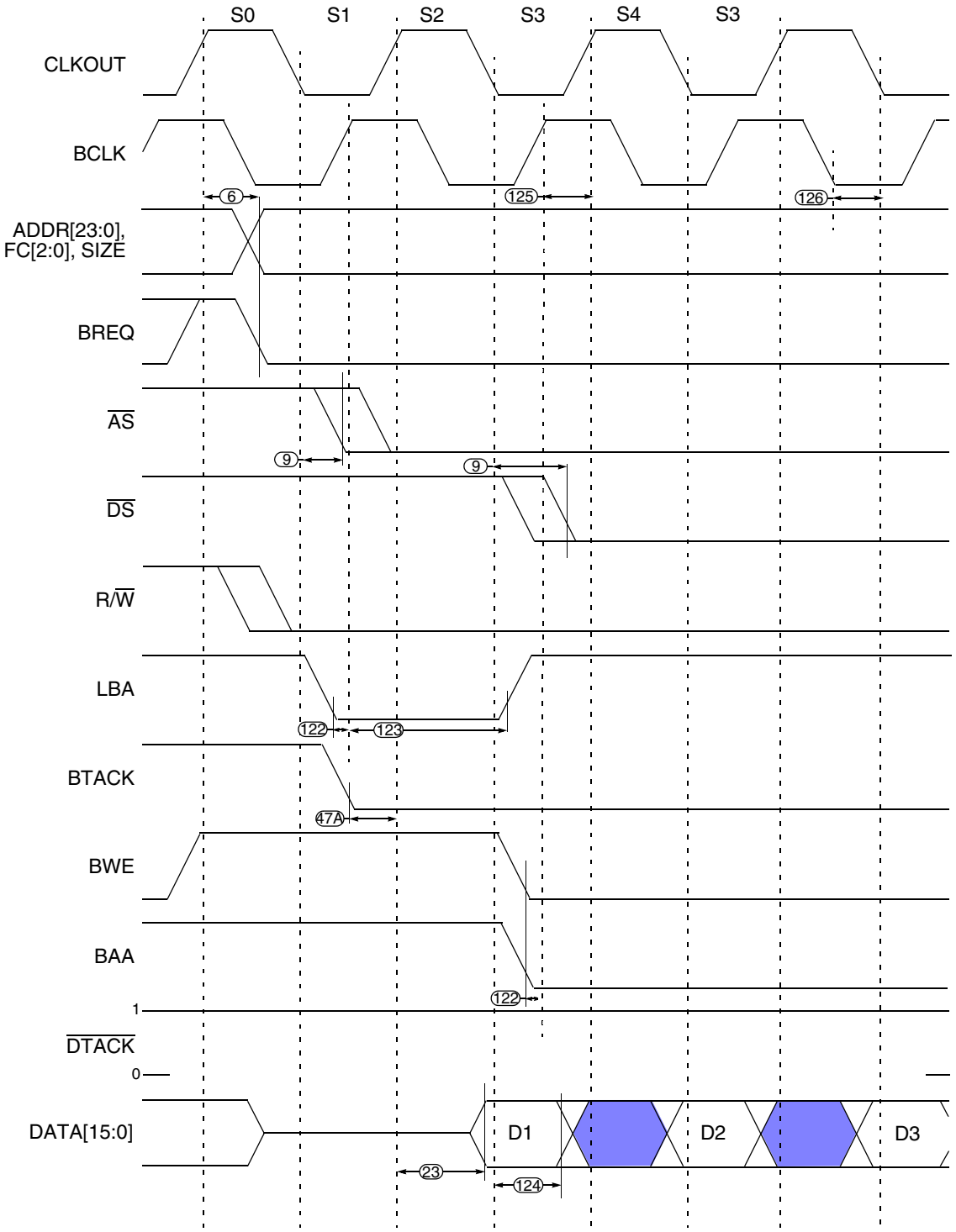


Figure E-13 Reset and Mode Select

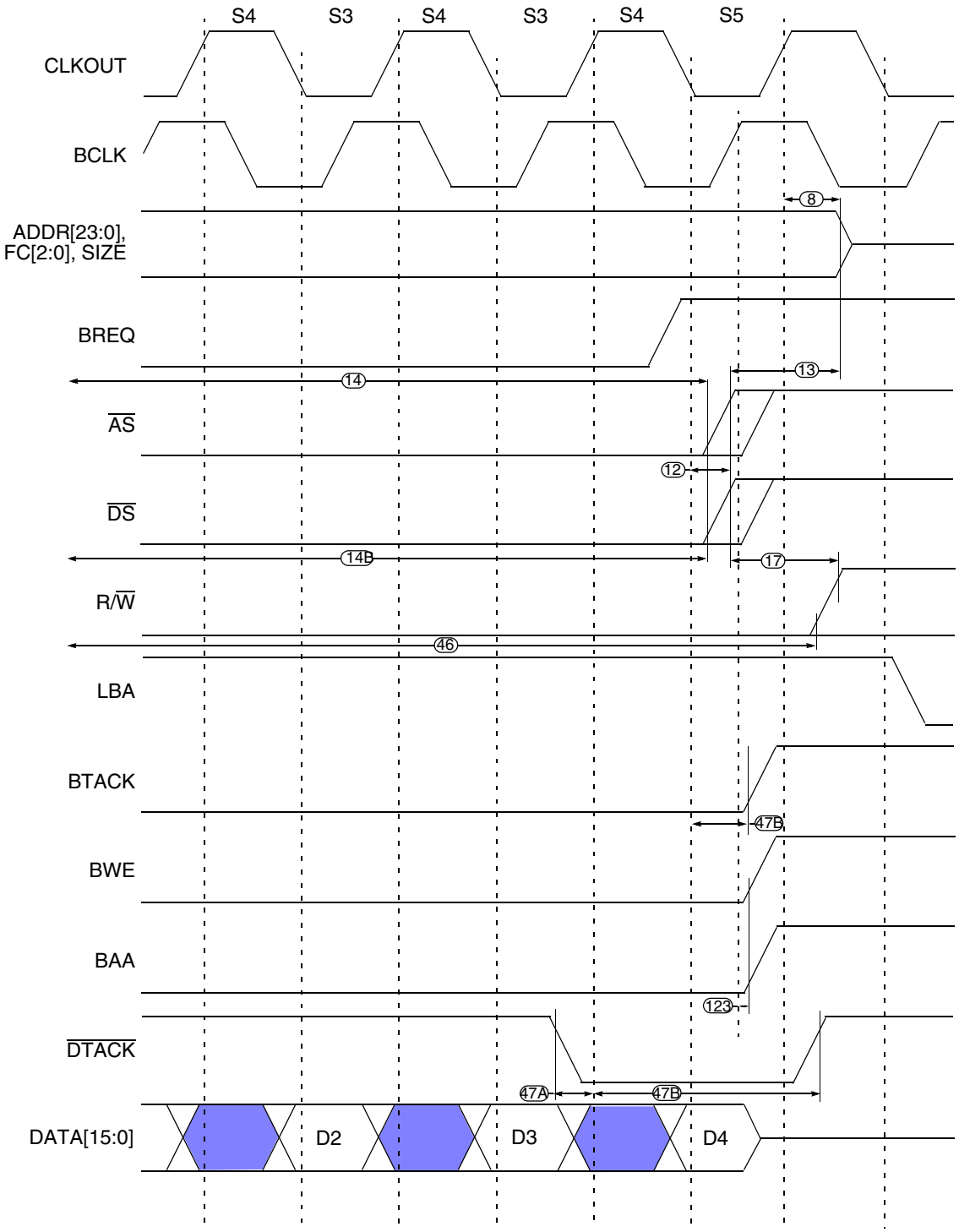
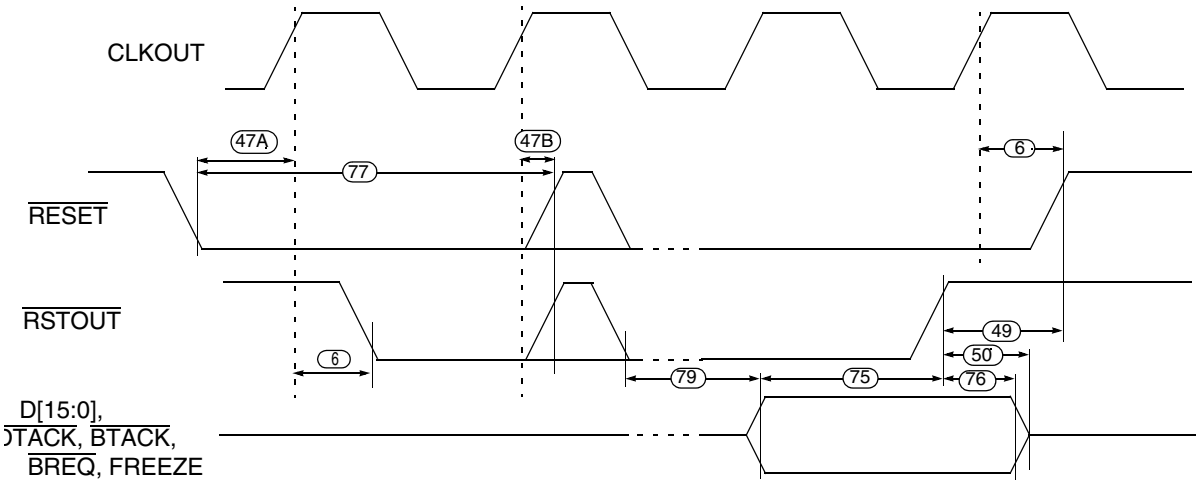


Figure E-14 Burst Write Cycle Timing — Initiation



Note: External data is sampled on the rising edge of CLOCKOUT
RESET and RSTOUT are driven on the rising edge of CLOCKOUT

Figure E-15 Burst Write Cycle Timing — Completion

Table E-5 BIM 25 MHz AC Timing

Num	Characteristic	Symbol	Min	Max	Unit
F1	Frequency of Operation, Fast Reference (5.44 MHz crystal)[1]	f_{FST}	(0.032)TBD	25	MHz
1	Clock Period	t_{CYC}	40	—	ns
1B	External Clock Input Period	t_{XCYC}	40	—	ns
2, 3	Clock Pulse Width	t_{CW}	0.5 t_{cyc-2}	—	ns
4, 5	EXTAL input Rise and Fall Time	t_{Crf}	—	2	ns
4A, 5A	Other Signal Rise and Fall Time	t_{Crf}	—	3	ns
6	Clock High to BREQ, RESET, RSTOUT Valid	t_{CHAV}	—	16	ns
6A	Clock High to Address, FC, SIZE, R/W, Valid	t_{CHAVA}	—	8	ns
6B	Address Valid to BCLK High (Including Pipeline Mode)	—	20	—	ns
7	Clock High to Address, Data, AS, DS, FC, SIZE, R/W, BREQ High Impedance	t_{CHAZx}	—	1.0	t_{cyc}
8	Clock High to Address, FC, SIZE, R/W Invalid	t_{CHAZn}	2	—	ns
8A	Clock High to BREQ Invalid	t_{CHAB}	2	20	ns
8B	BCLK High to Address Invalid (Pipeline Mode Only)	—	2	—	ns
9[2]	Clock Low to CS Valid (CS set to DS timing)	t_{CLSA}	2	16	ns
9A	Clock Low to AS, DS Valid	t_{STSA}	2	14	ns
9B	BCLK Low to CS Asserted (CS set to CE timing)	t_{STSB}	2	16	ns
11	Address, FC, SIZE, BREQ, R/W Valid to AS, DS, CS (CS set to DS timing) Valid	t_{AVSA}	7	—	ns
12	Clock Low to AS, DS, CS Invalid	t_{CLSN}	10	20	ns

Table E-5 BIM 25 MHz AC Timing (Continued)

Num	Characteristic	Symbol	Min	Max	Unit
13	AS, DS, CS Negated to Address, FC, SIZE Invalid (Address/ control Hold)	t_{SNAI}	7	—	t_{cyc}
14	AS Width Asserted, Read or Write Cycle[3] DS Width Asserted, Read Cycle	t_{SWA}	0.75 + ws	—	t_{cyc}
14A	CS Width Asserted, Read (CS set to CE timing)[3]	tSWAW	1 + ws	—	t_{cyc}
14B	DS, CS Width Asserted, Write (CS set to DS timing)[3]	tSWDW	ws - 0.25	—	t_{cyc}
15	AS, DS, CS Width Negated	t_{SN}	12	—	ns
17	AS, DS, CS Negated to R/W Invalid	t_{SNRN}	7	—	ns
22	R/W Low to CS Valid (CS set to CE timing)	t_{RASA}	7	—	ns
23	Clock High to Data-Out Valid, Write	t_{CHDO}	—	16	ns
25	DS, CS Negated to Data-Out Invalid (Data-Out Hold)	t_{SNDOI}	1	—	ns
26	Data-Out Valid to DS, CS Valid (Write)[4]	t_{DVSA}	7	—	ns
27A	Data-In Valid to BCLK High (Data Setup)	tCDSUA	7	—	ns
28	AS, DS Invalid to DTACK Invalid	t_{SNDN}	0	0.75	t_{cyc}
29	DS, CS Invalid to Data-In valid/High Impedance (Data-In Hold)[5]	t_{SNDI}	3	—	ns
30A	BCLK High to Data-In Invalid/High Impedance (Data-In Hold)	t_{CLDIA}	3	—	ns
46	R/$\overline{W}$ Width Asserted (Write or Read Cycle)[3]	t_{RWA}	1.5 + ws	—	t_{cyc}
47A	Asynchronous Input Setup Time $\overline{EBR}$, $\overline{DTACK}$, $\overline{BERR}$, $\overline{BTACK}$, $\overline{BKPT}$, $\overline{RESET}$	t_{AIST}	8	—	ns
47B	Asynchronous Input Hold Time $\overline{DTACK}$, $\overline{BERR}$, $\overline{BTACK}$, $\overline{BKPT}$, $\overline{RESET}$	t_{AIHT}	4	—	ns
49	$\overline{RSTOUT}$ High to $\overline{RESET}$ High	t_{IHRH}	12	—	ns
50	$\overline{RSTOUT}$ High to Mode Select Invalid/High Impedance	—	—	14	t_{cyc}
51	$\overline{RESET}$ High to $\overline{RSTOUT}$ Valid	t_{IHMZ}	—	0.5	t_{cyc}
53	Data-Out Hold from Clock Low	t_{DOCH}	0	—	ns
54	Clock Low to Data-Out High Impedance	t_{CHDH}	—	0.5	t_{cyc}
55	R/W Asserted to Data Bus Impedance Change	t_{RADC}	0.5	—	t_{cyc}
56	$\overline{RSTOUT}$ Pulse Width (Reset Instruction)	t_{RPWI}		256	t_{cyc}
56A	$\overline{RSTOUT}$ Pulse Width (Reset Pin)	t_{RPWP}	512	—	t_{cyc}
71	Data Valid to Clock Low (Show data setup)	t_{SCLDS}	5	—	ns
72	Clock Low to Data Invalid (Show data hold)	t_{SCLDH}	5	—	ns
75	Mode Select Setup Time	t_{MSS}	20	—	t_{cyc}
76	Mode Select Hold Time	t_{MSH}	0	—	ns
77	$\overline{RESET}$ (Input) Assertion Time[6]	t_{RSTA}	4	—	t_{cyc}
79	RESET Low and RSTOUT Low to Mode Select Drive	t_{RILM}	0	1	t_{cyc}
114	CSE0, CSE1 Valid to Clock High (Emulation, Setup)	t_{CECH}	5	—	ns
115	Clock High to CSE0, CSE1, FREEZE Invalid (Emulation, Hold)	t_{CHCEI}	5	—	ns
117	FREEZE Valid to Clock High (Emulation, Setup)	t_{FCH}	15	—	ns
122	LBA, BAA Valid to BCLK High	tBSU	24	—	ns
122A	HPCE Valid to BCLK High (LBA or BAA set to HPCE function)	tHPCE	30	—	ns
122B	BOE, BWE Valid to BCLK High (Burst, Setup)	tBSUB	0	—	ns
123	BCLK High to LBA, BAA, Invalid (Burst, Hold)	t_{BH}	2	16	ns
123A	BCLK High to BOE, BWE Invalid (Burst, Hold)	t_{BHA}	2	10	t_{cyc}

Table E-5 BIM 25 MHz AC Timing (Continued)

Num	Characteristic	Symbol	Min	Max	Unit
125	Clock High to BCLK High	tBCH	$0.25t_{cyc}$-1	$0.25t_{cyc}$+1	ns
126	Clock Low to BCLK Low	tBCL	$0.25t_{cyc}$-1	$0.25t_{cyc}$+1	ns
127	Clock High to $\overline{DS}$ Invalid(Show)	t_{CHDN}	2	16	ns
128	Clock High to Digital Output Valid	t_{CDIO}	90	600	ns

NOTES:
1. Minimum system clock frequency is crystal frequency divided by 128, subject to specified limits.
2. DS and CS do not assert on a zero wait state write cycle.
3. Number of wait states = ws. In a 2-clock bus cycle, ws =0. For 3-clock bus cycle, ws = 1.
4. Specification 26 is valid only for 3-clock or more bus cycles. Specification 26 does not apply to 2-clock bus cycles. Data-Out to CS asserted apply only when CS is configured as DS timing (STRB=1)
5. These hold times are specified with respect to DS or CS on asynchronous reads or CLKOUT. The user is free to choose either hold time.

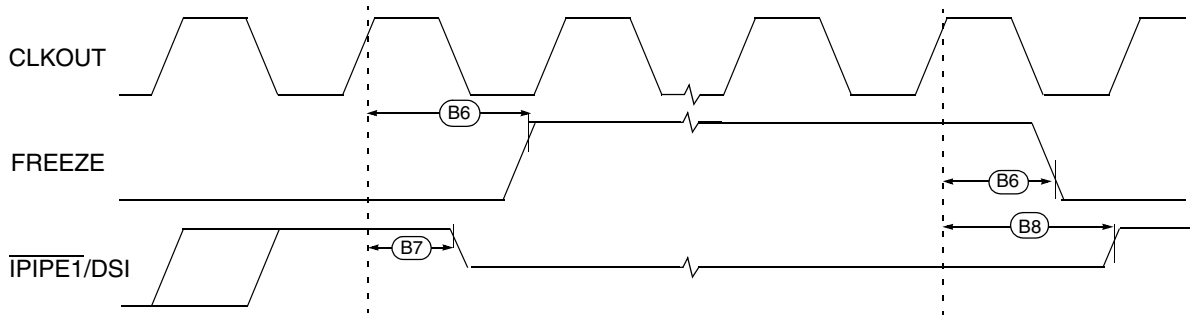


Figure E-16 Background Debug Mode (Freeze Assertion)

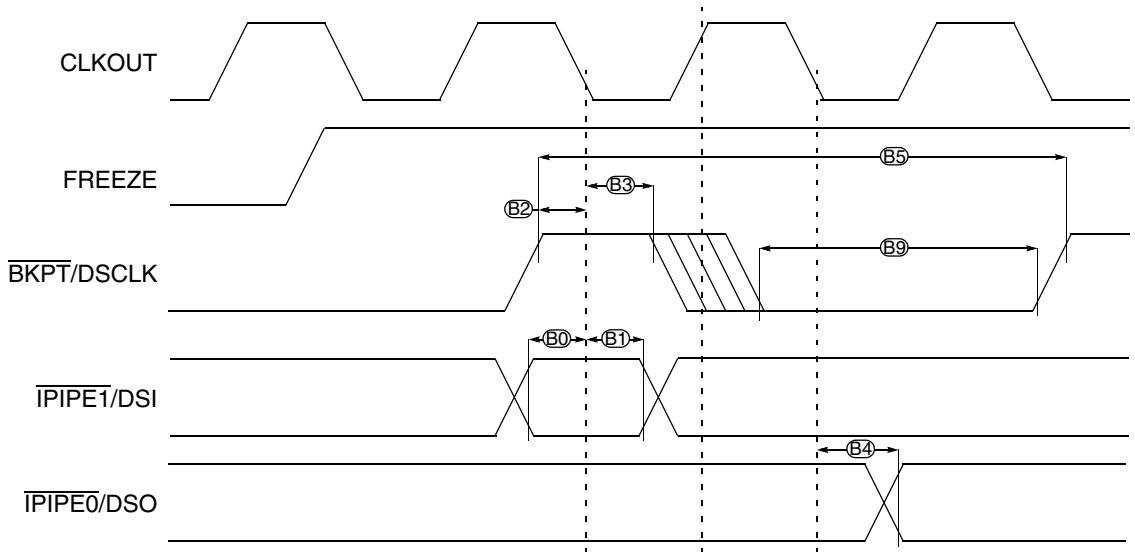


Figure E-17 Background Mode (Serial Communication)

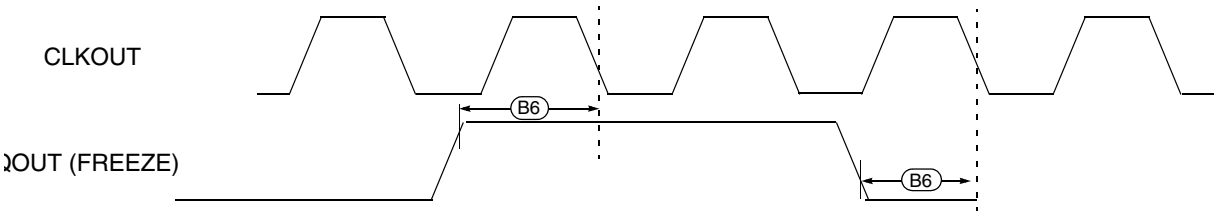
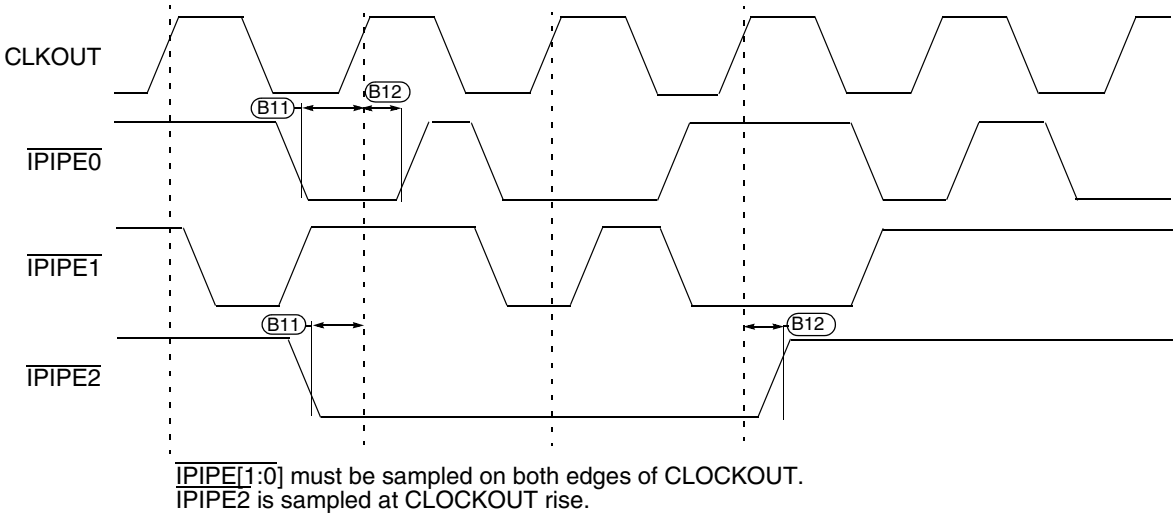


Figure E-18 QOUT Assertion



IPIPE[1:0] must be sampled on both edges of CLOCKOUT.
IPIPE2 is sampled at CLOCKOUT rise.

Figure E-19 Pipe Tracking Pin

Table E-6 Background Debug Mode Timing

Num	Characteristic	Symbol	Min	Max	Unit
B0	DSI Input Setup Time	t_{DSISU}	0.25	—	t_{cyc}
B1	DSI Input Hold Time	t_{DSIH}	0.25	—	t_{cyc}
B2	DSCLK Setup Time	t_{DSCSU}	0.25	—	t_{cyc}
B3	DSCLK Hold Time	t_{DSCH}	0.25	—	t_{cyc}
B4	DSO Delay Time	t_{DSOD}	—	0.5	t_{cyc}
B5	DSCLK Cycle Time	t_{DSCCYC}	2	—	t_{cyc}
B6	CLKOUT High to FREEZE Asserted/Negated	t_{FRZAN}	—	1	t_{cyc}
B6B	QOUT(FREEZE) Asserted/Negated to CLKOUT Low	t_{FRZAL}	4	t_{cyc}- 2	ns
B7	CLKOUT High to IPIPE1 High Impedance	t_{IFZ}	—	1	t_{cyc}
B8	CLKOUT High to IPIPE1 Valid	t_{IF}	—	1	t_{cyc}
B9	DSCLK Low Time	t_{DSCLO}	1	—	t_{cyc}
B11	IPIPE Valid to CLKOUT High or Low	tISU	2.5		ns
B12	CLKOUT High or Low to IPIPE Invalid	tIH	2		ns

E.2.3 FASRAM AC Timings

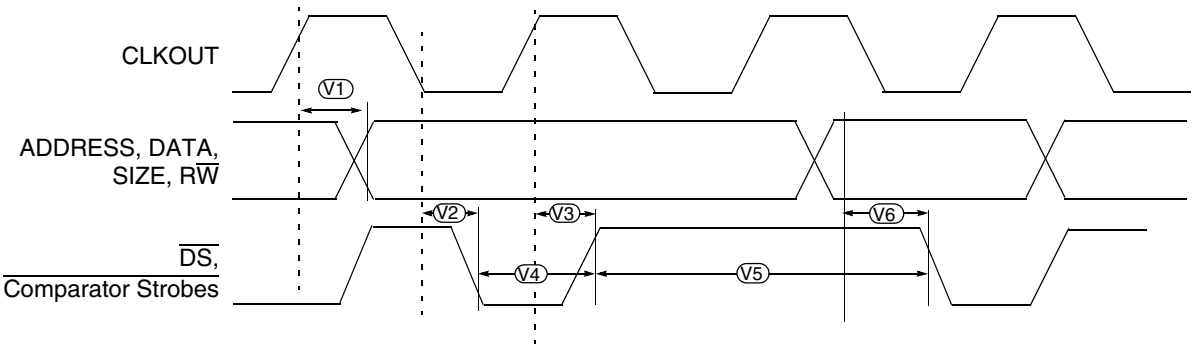


Figure E-20 Visibility Bus

Table E-7 Visibility Bus Timing

Num	Characteristic	Symbol	Min	Max	Unit
V1	CLKOUT High to Address, Data, Size, RW Valid	t_{VSU}	2	13	ns
V2	CLKOUT Low to STROBEs Asserted	t_{VH}	2	11.5	ns
V3	CLKOUT High to STROBEs Negated	t_{VSSU}	2	11.5	ns
V4	Strobe Width Asserted	t_{VSWA}	0.25	0.5	t_{cyc}
V5	Strobe Width Negated	t_{VSWN}	0.25	0.5	t_{cyc}
V6	Address, Data, R/W, SIZE Valid to Strobe Asserted	t_{VSH}	0.25	—	t_{cyc}

E.2.4 QADC64 Electrical and AC Characteristics

Table E-8 QADC64 Maximum Ratings

Num	Characteristic	Symbol	Min	Max	Unit
1	Analog Supply, with reference to V_{SSA}	V_{DDA}	-0.3	6.5	V
2	Internal Digital Supply, with reference to V_{SSI}	V_{DDI}	-0.3	6.5	V
3	Reference Supply, with reference to V_{RL}	V_{RH}	-0.3	6.5	V
4	V_{SS} Differential Voltage	$V_{SSI} - V_{SSA}$	-0.1	0.1	V
5	V_{DD} Differential Voltage	$V_{DDI} - V_{DDA}$	-6.5	6.5	V
6	V_{REF} Differential Voltage	$V_{RH} - V_{RL}$	-6.5	6.5	V
7	V_{RH} to V_{DDA} Differential Voltage	$V_{RH} - V_{DDA}$	-6.5	6.5	V
8	V_{RL} to V_{SSA} Differential Voltage	$V_{RL} - V_{SSA}$	-6.5	6.5	V
9	Disruptive Input Current[1, 2, 3, 4, 5, 6] $V_{NEGCLMAP} \cong -0.3$ V $V_{POSCLAMP} \cong V_{DDA} + 2$	I_{NA}	-500	500	µA
10	Adjacent Pin Attenuation[1, 5, 6, 7]	—	500	—	—
11	Maximum Input Current [3, 4, 6] $V_{NEGCLAMP} \cong -0.3$ V $V_{POSCLAMP} \cong V_{DDA} + 2$	I_{MA}	-25	25	mA

NOTES:

1. Below disruptive current conditions, the channel being stressed has conversion values of $3FF for analog inputs greater than V_{RH} and $000 for values less than V_{RL}. This assumes that $V_{RH} \leq V_{DDA}$ and $V_{RL} \geq V_{SSA}$ due to the presence of the sample amplifier. Other channels are not affected by non-disruptive conditions.
2. Input signals with large slew rates or high frequency noise components cannot be converted accurately. These signals also affect the conversion accuracy of other channels.
3. Exceeding limit may cause conversion error on stressed channels and on unstressed channels. Transitions within the limit do not affect device reliability or cause permanent damage.
4. Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values using positive and negative clamp values, then use the larger of the calculated values.
5. This parameter is periodically sampled rather than 100% tested.
6. Condition applies to one pin at a time only.
7. The attenuation factor reflects current induced on pins adjacent pins to the pin under negative stress conditions. A voltage drop may occur across the external source impedances of the adjacent pins impacting conversions on these pins.

Table E-9 QADC64 Electrical Characteristics (Operating)

Num	Parameter	Symbol	Min	Max	Unit
1	Analog Supply[1]	V_{DDA}	4.5	5.5	V
2	Internal Digital Supply[1]	V_{DDL}	3.0	3.6	V
3	V_{SS} Differential Voltage	$V_{SS} - V_{SSA}$	-100	100	mV
4	Reference Voltage Low[2]	V_{RL}	V_{SSA}	V_{SSA} + 0.1	V
5	Reference Voltage High[2]	V_{RH}	V_{DDA} - 0.1	V_{DDA}	V
6	V_{REF} Differential Voltage	$V_{RH} - V_{RL}$	4.5	5.5	V
7	Input Voltage	V_{INDC}	V_{SSA} - 0.3	V_{DDA} + 0.3	V
8	Input High Voltage, PQA and PQB	V_{IH}	0.7 (V_{DDA})	V_{DDA} + 0.3	V
9	Input Low Voltage, PQA and PQB	V_{IL}	V_{SSA} - 0.3	0.4 (V_{DDA})	V
10	Input Hysteresis, PQA, PQB[3]	V_{HYS}	0.5	—	V
11	Output Voltage, PQA[4] I_{OL} = 3.2 mA I_{OH}= -2.0 mA	 V_{OL} V_{OH}	 — 3.8	 0.45 —	 V V
12	Analog Supply Current Normal Operation[5] Low-Power Stop	 I_{DDA}	 — —	 5.0 10.0	 mA µA
13	Reference Supply Current, DC Reference Supply Current, Transient	I_{REF} i_{REF}	— —	250 2.0	µA mA
14	Load Capacitance, PQA output	C_L	—	50	pF
15	Input Current, Channel Off[6] PQA, PQB	I_{OFF}	-200	200	nA
16	Total Input Capacitance[7] PQA, PQB, Not Sampling Incremental Cap added during Sampling	C_{IN}	— —	15 5	pF
17	Disruptive Input Injection Current[8]	I_{INJ}	-3	3	mA

NOTES:

1. Refers to operation over full temperature and frequency range. Here V_{DDL}= 3.3 V typical.
2. To obtain full-scale, full-range results, $V_{SSA} \le V_{RL} \le V_{INDC} \le V_{RH} \le V_{DDA}$.
3. Parameter applies to the following pins:
   Port A: PQA[7:0]/AN[59:58]/ETRIG[2:1]
   Port B: PQB[7:0]/AN[3:0]/AN[51:48]/AN[Z:W]
4. Full driver (push-pull).
5. Current measured at maximum system clock frequency with QADC active.
6. Maximum leakage occurs at maximum operating temperature. Current decreases by approximately one-half for each 8 to 12 °C, in the ambient temperature range of 50 to 125 °C.
7. This parameter is periodically sampled rather than 100% tested.
8. Below disruptive current conditions, the channel being stressed has conversion values of $3FF for analog inputs greater than V_{RH} and $000 for values less than V_{RL}. This assumes that $V_{RH} \le V_{DDA}$ and $V_{RL} \ge V_{SSA}$ due to the presence of the sample amplifier. Other channels are not affected by non-disruptive conditions.

E.2.5 QADC64 AC TIMINGS

Table E-10 QADC64 Conversion Characteristics (Operating)

Num	Parameter	Symbol	Min	Typ	Max	Unit
1	QADC Clock (QCLK) Frequency[1]	F_{QCLK}	0.5	—	2.1	MHz
2	Conversion Cycles[2]	CC	14	—	28	QCLK cycles
3	Conversion Time [2,3] F_{QCLK} = 2.0MHz[1,4] Min = CCW/IST =%00 Max = CCW/IST =%11	T_{CONV}	7.0	—	14.0	µs
4	Stop Mode Recovery Time	T_{SR}	—	—	10	µs
5	Resolution[5]	—	—	5	—	mV
7	Disruptive Input Injection Current[7, 6, 7, 8, 9]	I_{INJ}	-3		3	mA
8	Coupling Ratio [7 ,10,11, 12] PQA, PQB	K	—	—	10^{-4}	V/V or A/A
9	Incremental Error due to injection current All channels have same 10 KΩ < Rs < 100 KΩ[15] Channel under test has Rs = 10 KΩ, I_{INJ} = ± 3 mA[13,14]				±1.0 ±1.0	Counts Counts
10	Source impedance at input[13]	R_S	—	10	100	KΩ
11	Incremental Capacitance during Sampling[14]	C_{SAMP}	—	—	5	pF

NOTES:

1. Conversion characteristics may vary with F_{QCLK} rate, possibly causing reduced conversion accuracy at max F_{QCLK} rate.
2. This specification depends on the value of “BYP” and “IST” control bits in the active CCW.
3. Assumes that f_{SYS} = 40 MHz.
4. Assumes F_{QCLK} = 2.00 MHz, with clock prescaler values of:
   QACR0: PSH =%01011, PSA =%0, PSL =111)
   CCW: BYP =%0
5. At $V_{RH} - V_{RL}$ = 5.12 V, one count = 5 mV.
6. Below disruptive current conditions, the channel being stressed has conversion values of $3FF for analog inputs greater than V_{RH} and $000 for values less than V_{RL}. This assumes that $V_{RH} \le V_{DDA}$ and $V_{RL} \ge V_{SSA}$ due to the presence of the sample amplifier. Other channels are not affected by non-disruptive conditions.
7. Exceeding limit may cause conversion error on stressed channels and on unstressed channels. Transitions within the limit do not affect device reliability or cause permanent damage.
8. Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values using $V_{POSCLAMP}$ = VDDA + 0.3 V and $V_{NEGCLAMP}$ = –0.3 V, then use the larger of the calculated values.
9. Condition applies to two pins, adjacent to channel being converted.
10. Coupling Ratio, K, is defined as the ratio of the output current, I_{out}, measured on the pin under test to the injection current, I_{inj}, when both adjacent pins are overstressed with the specified injection current. $K = I_{out}/ I_{inj}$ The input voltage error on the channel under test is calculated as $Verr = I_{inj} * K * R_S$
11. Total Injection current is determined by the number of channels injecting (e.g. 15), external injection voltage ($V_{INJ} - V_{POSCLAMP}$, or $V_{INJ} - V_{NEGCLAMP}$) , and the external source impedance, Rs, wherein all input channels have the same values. To determine the error voltage on the converted channel, only the two adjacent channels are expected to contribute to the error voltage:
    $$V_{errj} = (V_{INJ} - V_{CLAMP}) * K*2$$
12. Performance expected with production silicon.

13. Maximum source impedance is application-dependent. Error resulting from pin leakage depends on junction leakage into the pin and on leakage due to charge-sharing with internal capacitance.
Error from junction leakage is a function of external source impedance and input leakage current. In the following expression, expected error in result value due to junction leakage is expressed in voltage (V_{errj}):

$$V_{errj} = R_S * I_{OFF}$$

where I_{OFF} is a function of operating temperature. (See **Table E-9**, note 6).
Charge-sharing leakage is a function of input source impedance, conversion rate, change in voltage between successive conversions, and the size of the filtering capacitor used. Error levels are best determined empirically. In general, continuous conversion of the same channel may not be compatible with high source impedance.
14. For a maximum sampling error of the input voltage ≤ 1LSB, then the external filter capacitor, $C_f \geq 1024 * C_{SAMP}$. The value of C_{SAMP} in the new design may be reduced, or increased slightly.

E.2.6 QSM Electrical Characteristics

Table E-11 QSM DC Electrical Specifications

Characteristic	Symbol	Rise/Fall Time (ns)	Max Load (pf)
Load Capacitance for QSM Output Pins	$QSMC_L$	41	200
		31	150
		21	100

E.2.7 QSM AC Timings

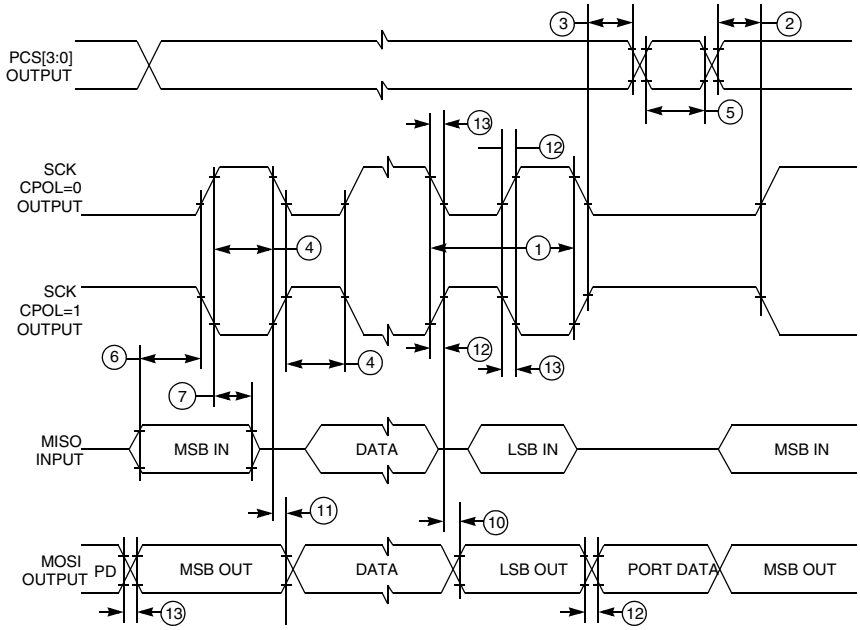


Figure E-21 QSPI Timing — Master, CPHA = 0

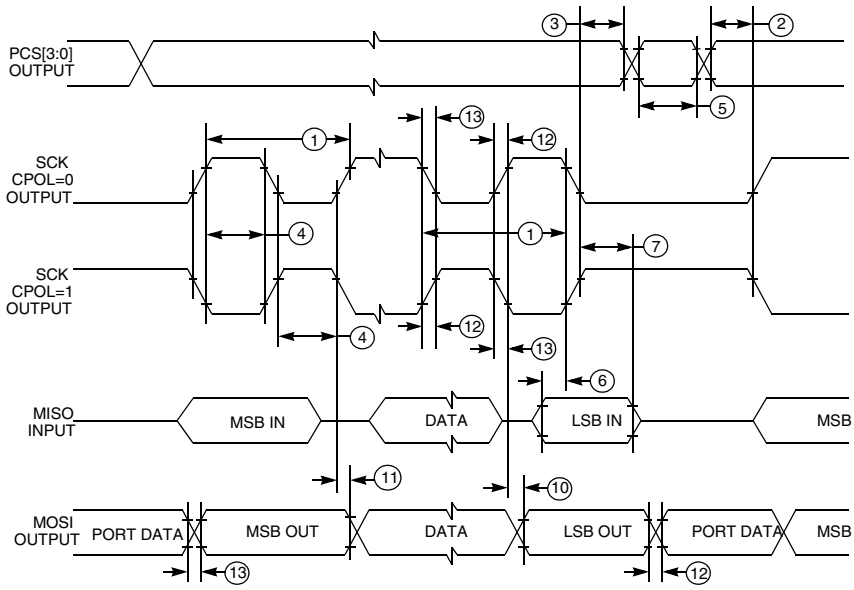


Figure E-22 QSPI Timing — Master, CPHA = 1

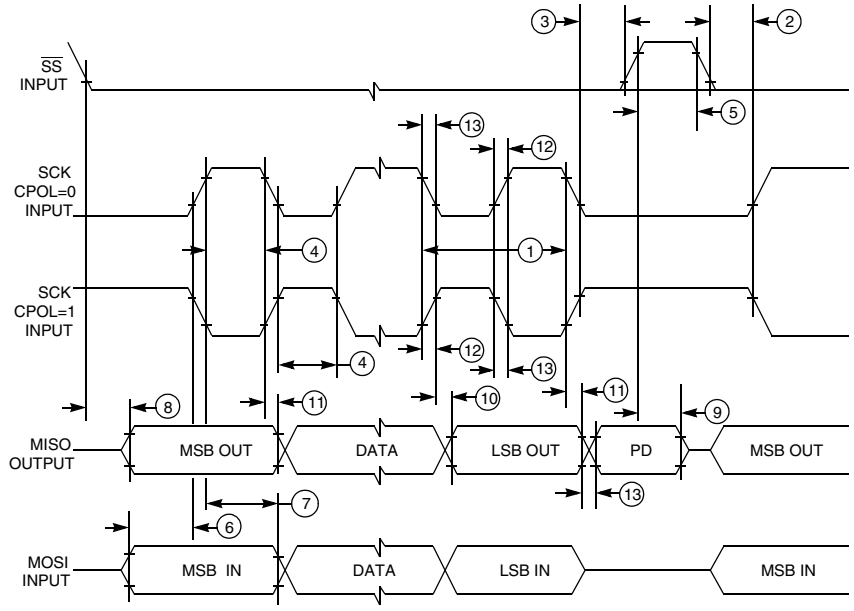


Figure E-23 QSPI Timing — Slave, CPHA = 0

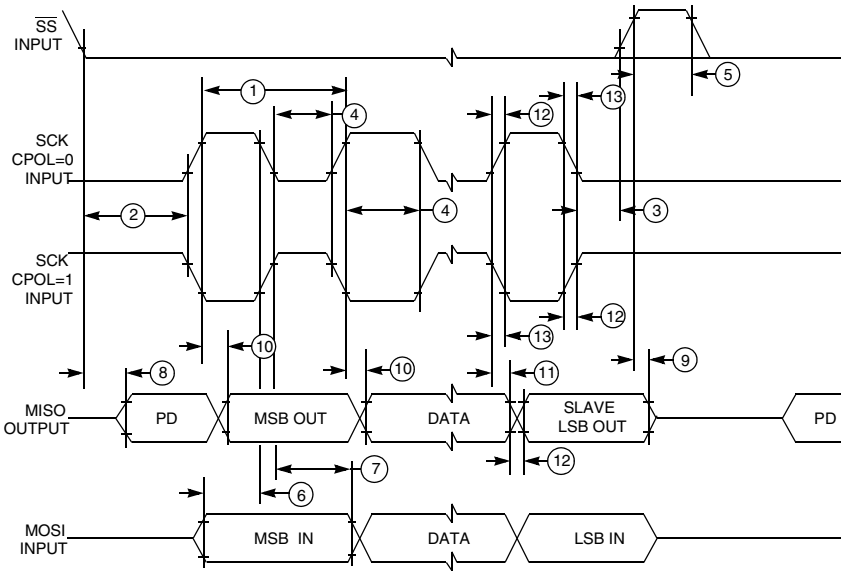


Figure E-24 QSPI TIming — Slave, CPHA = 1

Table E-12 QSPI Timing

Num	Function	Symbol	Min	Max	Unit
0	Operating Frequency Master Slave	f_{op}	DC DC	1/4 1/4	f_{sys} f_{sys}
1	Cycle Time Master Slave	t_{qcyc}	4 4	510 —	t_{cyc} t_{cyc}
2	Enable Lead Time Master Slave	t_{lead}	2 2	128 —	t_{cyc} t_{cyc}
3	Enable Lag Time Master Slave	t_{lag}	2 —	1/2 —	SCK t_{cyc}
4	Clock (SCK) High or Low Time Master Slave[1]	t_{sw}	2 tcyc – 60 2 tcyc – n	255 tcyc —	ns ns
5	Sequential Transfer Delay Master Slave (Does Not Require Deselect)	t_{td}	17 13	8192 —	t_{cyc} t_{cyc}
6	Data Setup Time (Inputs) Master Slave	t_{su}	30 20	— —	ns ns
7	Data Hold Time (Inputs) Master Slave	t_{hi}	0 20	— —	ns ns
8	Slave Access Time	t_a	—	1	t_{cyc}

Table E-12 QSPI Timing (Continued)

Num	Function	Symbol	Min	Max	Unit
9	Slave MISO Disable Time	t_{dis}	—	2	t_{cyc}
10	Data Valid (after SCK Edge) Master Slave	t_v	— —	50 50	ns ns
11	Data Hold Time (Outputs) Master Slave	t_{ho}	0 0	— —	ns ns
12	Rise Time Input Output	t_{ri} t_{ro}	— —	2 See Note 2	µs ns
13	Fall Time Input Output	t_{fi} t_{fo}	— —	2 See Note 2	µs ns

NOTES:
1. For high time, n = external SCK rise time; for low time, n = external SCK fall time.
2. The output rise time and fall time vary for different MCUs depending on the I/O pad used.

E.2.8 CTM9 AC Timing

Table E-13 MCSM Timing

Num	Parameter	Symbol	Min	Max	Unit
1	Input pin frequency[1]	f_{PCNTR}	0	$f_{sys}/4$	MHz
2	Input pin low time	t_{PINL}	$2.0/f_{sys}$	—	µs
3	Input pin high time	t_{PINH}	$2.0/f_{sys}$	—	µs
4	Clock pin to counter increment	t_{PINC}	$4.5/f_{sys}$	$6.5/f_{sys}$	µs
5	Clock pin to new TBB value	t_{PTBB}	$5.0/f_{sys}$	$7.0/f_{sys}$	µs
6	Clock pin to COF set ($FFFF)	t_{PCOF}	$4.5/f_{sys}$	$6.5/f_{sys}$	µs
7	Pin to IN bit delay	t_{PINB}	$1.5/f_{sys}$	$2.5/f_{sys}$	µs
8	Flag to IMB interrupt request	t_{FIRQ}	$1.0/f_{sys}$	$1.0/f_{sys}$	µs
9	Counter resolution[2]	t_{CRES}	—	$2.0/f_{sys}$	µs

NOTES:
1. Value applies when using external clock
2. Value applies when using internal clock. Minimum counter resolution depends on prescaler divide ratio selection

Table E-14 PWMSM Timing

Num	Parameter	Symbol	Min	Max	Unit
1	PWMSM output resolution[1]	t_{PWMR}	—	—	µs
2	PWMSM output pulse[2]	t_{PWMO}	$2.0/f_{sys}$	—	µs
3	PWMSM output pulse[3]	t_{PWMO}	$2.0/f_{sys}$	$2.0/f_{sys}$	µs
4	CPSM enable to output set PWMSM enabled before CPSM, DIV23 = 0 PWMSM enabled before CPSM, DIV23 = 1	t_{PWMP}	 $3.5/f_{sys}$ $6.5/f_{sys}$	—	µs
5	PWM enable to output set PWMSM enabled before CPSM, DIV23 = 0 PWMSM enabled before CPSM, DIV23 = 1	t_{PWME}	 $3.5/f_{sys}$ $5.5/f_{sys}$	 $4.5/f_{sys}$ $6.5/f_{sys}$	µs
6	FLAG to IMB interrupt request	t_{FIRQ}	$1.5/f_{sys}$	$2.5/f_{sys}$	µs

NOTES:
1. Minimum output resolution depends on counter and prescaler divide ratio selection.
2. Excluding the case where the output is always zero.
3. Excluding the case where the output is always zero.

Table E-15 SASM Timing

Num	Parameter	Symbol	Min	Max	Unit
1	Input pin low time	t_{PINL}	$2.0/f_{sys}$	—	µs
2	Input pin high time	t_{PINH}	$2.0/f_{sys}$	—	µs
3	Input capture resolution[1]	t_{RESCA}	—	$2.0/f_{sys}$	µs
4	Pin to input capture delay	t_{PCAPT}	$2.5/f_{sys}$	$4.5/f_{sys}$	µs
5	Pin to FLAG set	t_{PFLAG}	$2.5/f_{sys}$	$4.5/f_{sys}$	µs
6	Pin to IN bit delay	t_{PINB}	$1.5/f_{sys}$	$2.5/f_{sys}$	µs
7	OCT output pulse	t_{OCT}	$2.0/f_{sys}$	—	µs
8	Compare resolution[1]	t_{RESCM}	—	$2.0/f_{sys}$	µs
9	TBB change to FLAG set	t_{CFLAG}	$1.5/f_{sys}$	$1.5/f_{sys}$	µs
10	TBB change to pin change[2]	t_{CPIN}	$1.5/f_{sys}$	$1.5/f_{sys}$	µs
11	Flag to IMB interrupt request[2]	t_{FIRQ}	$1.0/f_{sys}$	$1.0/f_{sys}$	µs

NOTES:
1. Minimum resolution depends on counter and prescaler divide ratio selection.
2. Time given from when new value is stable on time base bus.

Table E-16 DASM Timing

Num	Parameter	Symbol	Min	Max	Unit
1	Input pin low time	t_{PINL}	$2.0/f_{sys}$	—	µs
2	Input pin high time	t_{PINH}	$2.0/f_{sys}$	—	µs
3	Input capture resolution[1]	t_{RESCA}	—	$2.0/f_{sys}$	µs
4	Pin to input capture delay	t_{PCAPT}	$2.5/f_{sys}$	$4.5/f_{sys}$	µs
5	Pin to FLAG set	t_{PFLAG}	$2.5/f_{sys}$	$4.5/f_{sys}$	µs
6	Pin to IN bit delay	t_{PINB}	$1.5/f_{sys}$	$2.5/f_{sys}$	µs
7	OCT output pulse	t_{OCT}	$2.0/f_{sys}$	—	µs
8	Compare resolution[1]	t_{RESCM}	—	$2.0/f_{sys}$	µs
9	TBB change to FLAG set	t_{CFLAG}	$1.5/f_{sys}$	$1.5/f_{sys}$	µs
10	TBB change to pin change[2]	t_{CPIN}	$1.5/f_{sys}$	$1.5/f_{sys}$	µs
11	Flag to IMB interrupt request[2]	t_{FIRQ}	$1.0/f_{sys}$	$1.0/f_{sys}$	µs

NOTES:
1. Minimum resolution depends on counter and prescaler divide ratio selection.
2. Time given from when new value is stable on time base bus.

Table E-17 PWMSM Timing

Num	Parameter	Symbol	Min	Max	Unit
1	PWMSM output resolution[1]	t_{PWMR}	—	—	µs
2	PWMSM output pulse[2]	t_{PWMO}	$2.0/f_{sys}$	—	µs
3	PWMSM output pulse[2]	t_{PWMO}	$2.0/f_{sys}$	$2.0/f_{sys}$	µs
4	CPSM enable to output set PWMSM enabled before CPSM, DIV23 = 0 PWMSM enabled before CPSM, DIV23 = 1	t_{PWMP}	 $3.5/f_{sys}$ $6.5/f_{sys}$	—	µs
5	PWM enable to output set PWMSM enabled before CPSM, DIV23 = 0 PWMSM enabled before CPSM, DIV23 = 1	t_{PWME}	 $3.5/f_{sys}$ $5.5/f_{sys}$	 $4.5/f_{sys}$ $6.5/f_{sys}$	µs
6	FLAG to IMB interrupt request	t_{FIRQ}	$1.5/f_{sys}$	$2.5/f_{sys}$	µs

NOTES:
1. Minimum output resolution depends on counter and prescaler divide ratio selection.
2. Excluding the case where the output is always zero.

E.2.9 TPU AC Characteristics

Table E-18 Time Processor Unit Timing (TPU)

Num	Rating	Symbol	Min	Max	Unit
1	CLKOUT High to TPU2 Output Channel Valid	t_{CHTOV}	2	23	ns
2	CLKOUT High to TPU2 Output Channel Hold	t_{CHTOH}	0	20	ns
3	TPU2 Input Channel Pulse Width	t_{TIPW}	4	—	t_{cyc}

E.2.10 TouCAN AC Characteristics

Table E-19 TouCAN AC Characteristics

Num	Parameter	Symbol	Value	Unit
1	CNTX0 – Delay from ICLOCK	—	19	ns
2	CNRX0 – Set-up to rise ICLOCK	—	0	ns
3	TOUCAN serial (Tx, Rx pins) max frequency	—	1	MHz

E.2.11 DLCMD3 AC Characteristics

Table E-20 DLCMD2 Requirements

Operating Temperature	IC Package	Clock Speed[1]	Quiescent Current Draw
–40 to +125 °C	NA	2.00 Mhz (Min) to 128.00 MHz (Max)	50 µA LPSTOP, 200 µA idle

NOTE:

1. No additional clock part needed for the DLCMD2. The DLCMD2 will adapt to the CPU (IMB3) clocking using software programmable prescaler. DLCMD2 must correctly operate over any clock jitter present within the integrated device.