

# MC68CK338

## *Technical Summary*

# 32-Bit Modular Microcontroller

### 1 Introduction

The MC68CK338, a highly-integrated 32-bit microcontroller, combines high-performance data manipulation capabilities with powerful peripheral subsystems. The MCU is built up from standard modules that interface through a common intermodule bus (IMB). Standardization facilitates rapid development of devices tailored for specific applications.

The MCU incorporates a low-power 32-bit CPU (CPU32L), a low-power system integration module (SIML), a queued serial module (QSM), and a configurable timer module 6 (CTM6).

The MCU clock can either be synthesized from an external reference or input directly. Operation with a 32.768 kHz reference frequency is standard. The maximum system clock speed is 14.4 MHz. System hardware and software allow changes in clock rate during operation. Because MCU operation is fully static, register and memory contents are not affected by clock rate changes.

High-density complementary metal-oxide semiconductor (HCMOS) architecture and 3V nominal operation make the basic power consumption of the MCU low. Power consumption can be minimized by either stopping the system clock, or alternatively, stopping the system clock only at the CPU32L, and allowing the other modules to continue operation. The CPU32 instruction set includes a low-power stop (LPSTOP) command that allows either of these power saving modes.

The CTM6 includes new features such as a port I/O submodule, a 64-byte RAM submodule and a real time clock submodule.

Refer to the Motorola Microcontroller Technologies Group Web page at <http://www.mcu.sps.mot.com> for the most current listing of device errata and customer information.

**Table 1 Ordering Information**

Package Type	Frequency (MHz)	Voltage	Temperature	Package Order Quantity	Order Number
144-Pin TQFP	14.4 MHz	2.7V to 3.6V	- 40 to + 85 °C	2 pc tray 60 pc tray 300 pc tray	SPMC68CK338CPV14 MC68CK338CPV14 MC68CK338CPV14B1

This document contains information on a new product. Specifications and information herein are subject to change without notice.



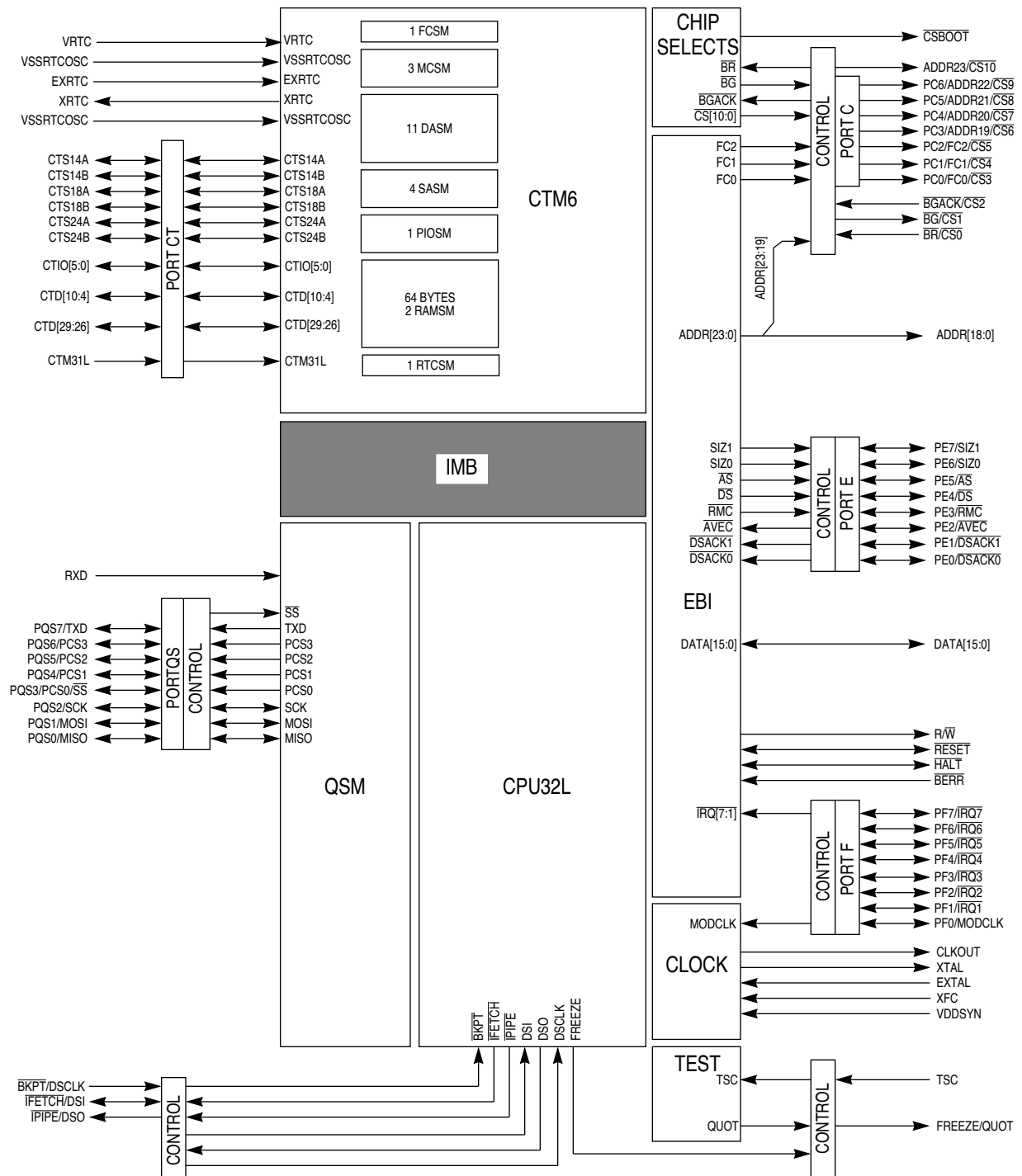
# TABLE OF CONTENTS

Section		Page
<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Features .....	3
1.2	Block Diagram .....	4
1.3	Pin Assignments .....	5
1.4	Address Map .....	6
1.5	Intermodule Bus .....	6
<b>2</b>	<b>Signal Descriptions</b>	<b>7</b>
2.1	Pin Characteristics .....	7
2.2	MCU Power Connections .....	8
2.3	MCU Driver Types .....	8
2.4	Signal Characteristics .....	9
2.5	Signal Function .....	10
<b>3</b>	<b>Low-Power System Integration Module</b>	<b>12</b>
3.1	Overview .....	12
3.2	System Configuration Block .....	14
3.3	System Clock .....	16
3.4	System Protection Block .....	22
3.5	External Bus Interface .....	26
3.6	Chip-Selects .....	30
3.7	General-Purpose Input/Output .....	38
3.8	Resets .....	41
3.9	Interrupts .....	44
3.10	Factory Test Block .....	47
<b>4</b>	<b>Low-Power Central Processor Unit</b>	<b>48</b>
4.1	Overview .....	48
4.2	Programming Model .....	48
4.3	Status Register .....	50
4.4	Data Types .....	51
4.5	Addressing Modes .....	51
4.6	Instruction Set Summary .....	51
4.7	Background Debugging Mode .....	56
<b>5</b>	<b>Queued Serial Module</b>	<b>57</b>
5.1	Overview .....	57
5.2	Address Map .....	58
5.3	Pin Function .....	59
5.4	QSM Registers .....	59
5.5	QSPI Submodule .....	64
5.6	SCI Submodule .....	72
<b>6</b>	<b>Configurable Timer Module 6</b>	<b>78</b>
6.1	Overview .....	78
6.2	Address Map .....	80
6.3	Time Base Bus System .....	82
6.4	Bus Interface Unit Submodule (BIUSM) .....	84
6.5	Counter Prescaler Submodule (CPSM) .....	85
6.6	Clock Sources for Counter Submodules .....	87
6.7	Free-Running Counter Submodule (FCSM) .....	87
6.8	Modulus Counter Submodule (MCSM) .....	90
6.9	Single Action Submodule (SASM) .....	93
6.10	Double-Action Submodule (DASM) .....	97
6.11	Real-Time Clock Submodule (RTCSM) with Low-Power Oscillator .....	104
6.12	Parallel Port I/O Submodule (PIOSM) .....	107
6.13	Static RAM Submodule (RAMSM) .....	108
6.14	RTCSM and RAMSM Standby Operation .....	108
6.15	CTM6 Interrupts .....	109
<b>7</b>	<b>Electrical Characteristics</b>	<b>111</b>

## 1.1 Features

- Modular Architecture
- Low-Power Central Processing Unit (CPU32L)
  - Virtual memory implementation
  - Loop mode of instruction execution
  - Improved exception handling for controller applications
  - Table lookup and interpolate instruction
  - CPU-only LPSTOP operation/normal MCU LPSTOP operation
- Low-Power System Integration Module (SIML)
  - External bus support
  - Twelve programmable chip-select outputs
  - System protection logic
  - On-chip PLL for system clock
  - Watchdog timer, clock monitor, and bus monitor
  - Expanded LPSTOP operation
- Queued Serial Module (QSM)
  - Enhanced serial communication interface (SCI)
  - Queued serial peripheral interface (QSPI)
  - Dual function I/O ports
- Configurable Timer Module 6 (CTM6)
  - One bus interface unit submodule (BIUSM)
  - One counter prescaler submodule (CPSM)
  - Three modulus counter submodules (MCSM)
  - One free-running counter submodule (FCSM)
  - Eleven double action submodules (DASM)
  - Four (eight channels) single action submodules (SASM)
  - One real time clock submodule (RTCSM)
  - One port I/O submodule (PIOSM)
  - Two 32-byte RAM submodules (RAMSM)

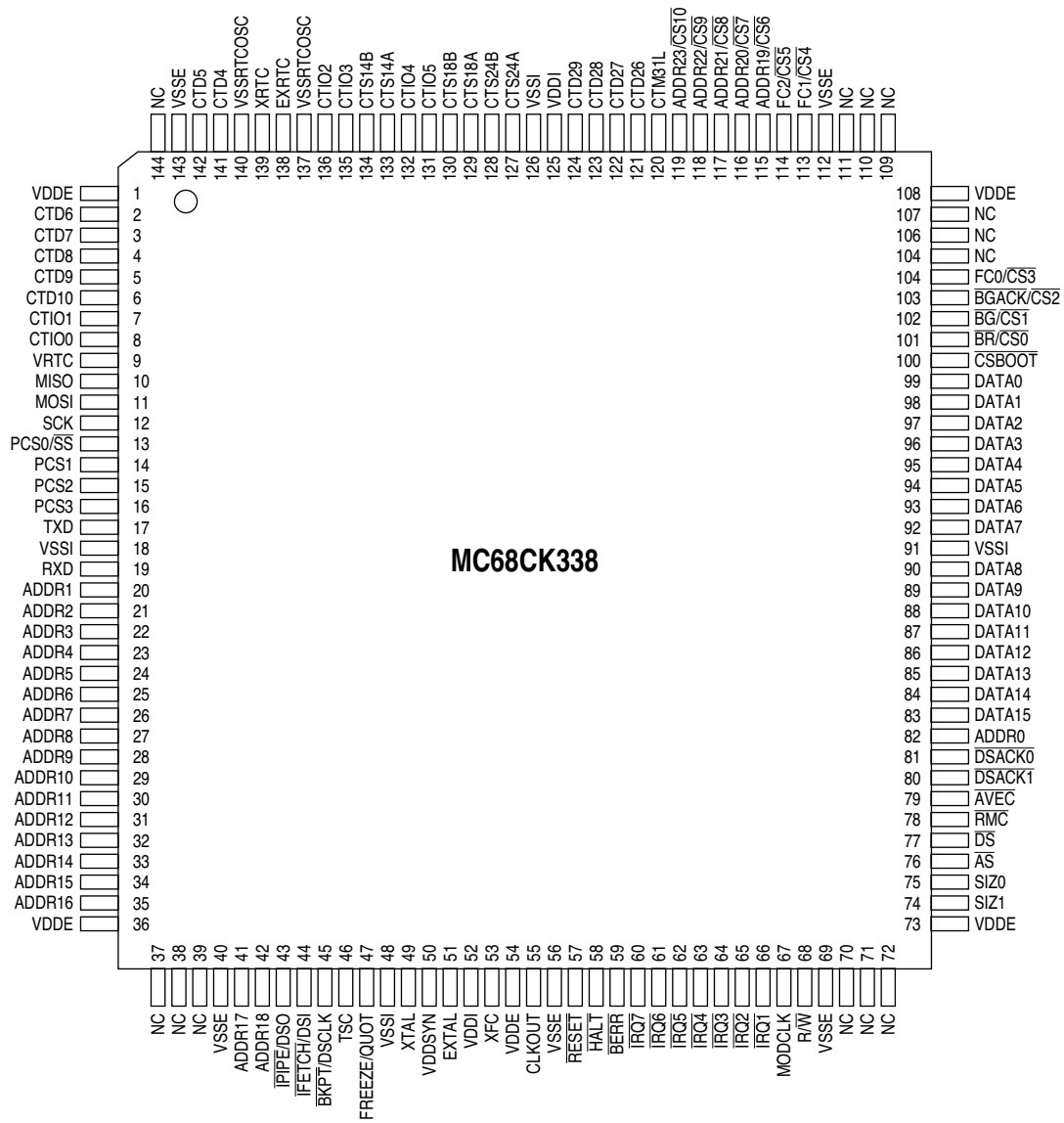
## 1.2 Block Diagram



338 BLOCK

Figure 1 MC68CK338 Block Diagram

### 1.3 Pin Assignments

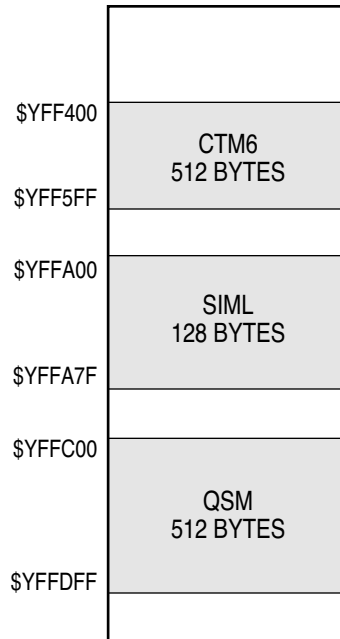


338 144-PIN QFP

Figure 2 MC68CK338 Pin Assignments

## 1.4 Address Map

Figure 3 shows a map of the MCU internal addresses. Unimplemented blocks are mapped externally.



Y = M111, WHERE M IS THE STATE OF THE MODULE MAPPING (MM) BIT IN THE SIML CONFIGURATION REGISTER.

338 ADDRESS MAP

Figure 3 MC68CK338 Address Map

## 1.5 Intermodule Bus

The IMB is a standardized bus developed to facilitate design and operation of modular microcontrollers. It contains circuitry that supports exception processing, address space partitioning, multiple interrupt levels, and vectored interrupts. The standardized modules in the MCU communicate with one another and with external components via the IMB. The IMB uses 24 address lines and 16 data lines.

## 2 Signal Descriptions

### 2.1 Pin Characteristics

**Table 2** shows MCU pins and their characteristics. All inputs detect CMOS logic levels. All inputs can be put in a high-impedance state, but the method of doing this differs depending upon pin function. Refer to **Table 4** for a description of output drivers. An entry in the discrete I/O column of **Table 2** indicates that a pin has an alternate I/O function. The port designation is given when it applies. Refer to **Figure 1** for information about port organization.

**Table 2 MCU Pin Characteristics**

Pin Mnemonic	Output Driver	Input Synchronized	Input Hysteresis	Discrete I/O	Port Designation
ADDR23/CS1 $\bar{0}$	A	Yes	No	—	—
ADDR[22:19]/CS[9:6]	A	Yes	No	O	PC[6:3]
ADDR[18:0]	A	Yes	No	—	—
$\bar{AS}$	B	Yes	No	I/O	PE5
$\bar{AVEC}$	B	Yes	No	I/O	PE2
$\bar{BERR}$	B	Yes <sup>1</sup>	No	—	—
BG/CS1	B	—	—	—	—
BGACK/CS2	B	Yes	No	—	—
BKPT/DSCLK	—	Yes	Yes	—	—
$\bar{BR}/CS0$	B	Yes	No	—	—
CLKOUT	A	—	—	—	—
$\bar{CSBOOT}$	B	—	—	—	—
CTD[29:26]	Ao	Yes	Yes	I/O	—
CTD[10:4]	Ao	Yes	Yes	I/O	—
CTIO[5:0]	A	Yes	Yes	I/O	—
CTM31L	A	Yes	Yes	I	—
CTS24[B:A]	A	Yes	Yes	I/O	—
CTS18[B:A]	A	Yes	Yes	I/O	—
CTS14[B:A]	A	Yes	Yes	I/O	—
DATA[15:0]	Aw	Yes <sup>2</sup>	No	—	—
$\bar{DS}$	B	Yes	No	I/O	PE4
$\bar{DSACK}[1:0]$	B	Yes	No	I/O	PE[1:0]
DSI/IFETCH	A	Yes	Yes	—	—
DSO/PIPE	A	—	—	—	—
EXRTC	—	—	Yes	—	—
EXTAL	—	—	Yes	—	—
FC[2:0]/CS[5:3]	A	Yes	No	O	PC[2:0]
FREEZE/QUOT	A	—	—	—	—
$\bar{HALT}$	Bo	Yes <sup>1</sup>	No	—	—
IRQ[7:1]	B	Yes	Yes	I/O	PF[7:1]
MISO	Bo	Yes <sup>2</sup>	Yes	I/O	PQS0
MODCLK	B	Yes <sup>2</sup>	No	I/O	PF0
MOSI	Bo	Yes <sup>2</sup>	Yes	I/O	PQS1

**Table 2 MCU Pin Characteristics (Continued)**

Pin Mnemonic	Output Driver	Input Synchronized	Input Hysteresis	Discrete I/O	Port Designation
PCS0/ $\overline{SS}$	Bo	Yes <sup>2</sup>	Yes	I/O	PQS3
PCS[3:1]	Bo	Yes <sup>2</sup>	Yes	I/O	PQS[6:4]
RESET	Bo	Yes	Yes	—	—
$\overline{RMC}$	A	Yes	Yes	I/O	PE3
R/ $\overline{W}$	A	Yes	No	—	—
RXD	—	No	Yes	—	—
SCK	Bo	Yes <sup>2</sup>	Yes	I/O	PQS2
SIZ[1:0]	B	Yes	No	I/O	PE[7:6]
TSC	—	Yes	Yes	—	—
TXD	Bo	Yes <sup>2</sup>	Yes	I/O	PQS7
XFC	—	—	—	—	—
XRTC	—	—	—	—	—
XTAL	—	—	—	—	—

NOTES:

1.  $\overline{HALT}$  and  $\overline{BERR}$  synchronized only if late  $\overline{HALT}$  or  $\overline{BERR}$ .
2. DATA[15:0] synchronized during reset only. MODCLK and QSM pins synchronized only if used as port I/O pins.

## 2.2 MCU Power Connections

**Table 3 MCU Power Connections**

$V_{DDSYN}$	Clock Synthesizer
$V_{DDE}, V_{SSE}$	External periphery power (source and drain)
$V_{DDI}, V_{SSI}$	Internal module power (source and drain)
$V_{RTC}$	RTCSM/RAMSM standby power
$V_{SSRTCOSC}$	Ground connection for real-time clock oscillator

## 2.3 MCU Driver Types

**Table 4 MCU Output Driver Types**

Type	Description
A	Output-only signals that are always driven; no external pull-up required
Ao	Type A output that can be operated in an open drain mode
Aw	Type A output with weak P-channel pull-up during reset
B	Three-state output that includes circuitry to pull up output before high impedance is established, to ensure rapid rise time. An external holding resistor is required to maintain logic level while the pin is in the high-impedance state.
Bw	Type B output with weak P-channel pull-up during reset
Bo	Type B output that can be operated in an open-drain mode



## 2.4 Signal Characteristics

**Table 5 MCU Signal Characteristics**

Signal Name	MCU Module	Signal Type	Active State
ADDR[23:0]	SIML	Bus	—
$\overline{AS}$	SIML	Output	0
$\overline{AVEC}$	SIML	Input	0
$\overline{BERR}$	SIML	Input	0
$\overline{BG}$	SIML	Output	0
$\overline{BGACK}$	SIML	Input	0
$\overline{BKPT}$	CPU32L	Input	0
BR	SIML	Input	0
CLKOUT	SIML	Output	—
$\overline{CS}$ [10:0]	SIML	Output	0
$\overline{CSBOOT}$	SIML	Output	0
CTD[29:26]	CTM6	Input/Output	—
CTD[10:4]	CTM6	Input/Output	—
CTIO[5:0]	CTM6	Input/Output	—
CTM31L	CTM6	Input	—
CTS24[B:A]	CTM6	Input/Output	—
CTS18[B:A]	CTM6	Input/Output	—
CTS14[B:A]	CTM6	Input/Output	—
DATA[15:0]	SIML	Bus	—
$\overline{DS}$	SIML	Output	0
$\overline{DSACK}$ [1:0]	SIML	Input	0
DSCCLK	CPU32L	Input	Serial Clock
DSI	CPU32L	Input	—
DSO	CPU32L	Output	—
EXRTC	CTM6	Input	—
EXTAL	SIML	Input	—
FC[2:0]	SIML	Output	—
FREEZE	SIML	Output	1
$\overline{HALT}$	SIML	Input/Output	0
$\overline{IFETCH}$	CPU32L	Output	—
$\overline{IPIPE}$	CPU32L	Output	—
$\overline{IRQ}$ [7:1]	SIML	Input	0
MISO	QSM	Input/Output	—
MODCLK	SIML	Input	—
MOSI	QSM	Input/Output	—
PC[6:0]	SIML	Output	—
PCS[3:0]	QSM	Input/Output	—
PE[7:0]	SIML	Input/Output	—
PF[7:0]	SIML	Input/Output	—
PQS[7:0]	QSM	Input/Output	—
QUOT	SIML	Output	—

**Table 5 MCU Signal Characteristics (Continued)**

Signal Name	MCU Module	Signal Type	Active State
$\overline{\text{RESET}}$	SIML	Input/Output	0
$\overline{\text{RMC}}$	SIML	Output	0
$\overline{\text{R/W}}$	SIML	Output	0
RXD	QSM	Input	—
SCK	QSM	Input/Output	—
SIZ[1:0]	SIML	Output	—
$\overline{\text{SS}}$	QSM	Input	0
TSC	SIML	Input	—
TXD	QSM	Output	—
XFC	SIML	Input	—
XRTC	CTM6	Output	—
XTAL	SIML	Output	—

## 2.5 Signal Function

**Table 6 MCU Signal Function**

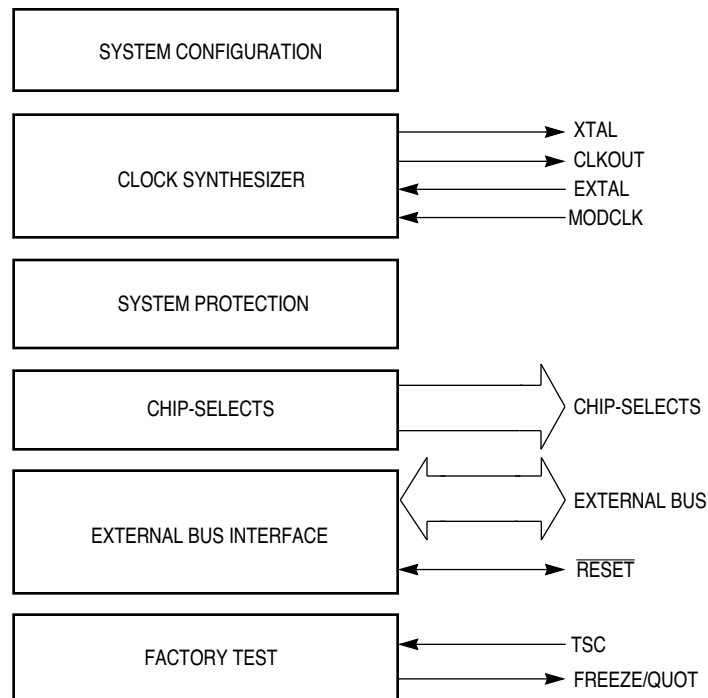
Signal Name	Mnemonic	Function
Address Bus	ADDR[23:0]	24-bit address bus
Address Strobe	$\overline{\text{AS}}$	Indicates that a valid address is on the address bus
Autovector	$\overline{\text{AVEC}}$	Requests an automatic vector during interrupt acknowledge
Bus Error	$\overline{\text{BERR}}$	Signals a bus error to the CPU
Bus Grant	$\overline{\text{BG}}$	Indicates that the MCU has relinquished the bus
Bus Grant Acknowledge	$\overline{\text{BGACK}}$	Indicates that an external device has assumed bus mastership
Breakpoint	$\overline{\text{BKPT}}$	Signals a hardware breakpoint to the CPU
Bus Request	$\overline{\text{BR}}$	Indicates that an external device requires bus mastership
System Clockout	CLKOUT	System clock output
Chip Selects	$\overline{\text{CS}}[10:0]$	Select external devices at programmed addresses
Boot Chip Select	$\overline{\text{CSBOOT}}$	Chip select for external boot startup ROM
Configurable Timer Double-Action	CTD[29:26], CTD[10:4]	Double-action submodule (DASM) signals. Can also be used as general purpose I/O pins
Configurable Timer Modulus Counter Load	CTM31L	External load for modulus counter. Can also be used as general purpose input
RTC Configurable Timer Oscillator	EXRTC, XRTC	CTM real time clock oscillator input/output
Configurable Timer Port Input/Output	CTIO[5:0]	General-purpose I/O pins
Configurable Timer Single-Action	CTS24[B:A] CTS18[B:A] CTS14[B:A]	Single-action submodule (SASM) signals. Can also be used as general purpose I/O pins
Crystal Oscillator	EXTAL, XTAL	Connections for clock synthesizer circuit reference; a crystal or an external oscillator can be used
Data Bus	DATA[15:0]	16-bit data bus

**Table 6 MCU Signal Function (Continued)**

Signal Name	Mnemonic	Function
Data Strobe	$\overline{DS}$	Indicates that an external device should place valid data on the data bus during a read cycle and that valid data has been placed on the bus by the CPU during a write cycle
Data and Size Acknowledge	$\overline{DSACK}[1:0]$	Acknowledges to the SIML that data has been received for a write cycle, or that data is valid on the data bus for a read cycle
Development Serial In, Out, Clock	DSI, DSO, DSCLK	Serial I/O and clock for background debugging mode
Function Codes	FC[2:0]	Identify processor state and current address space
Freeze	FREEZE	Indicates that the CPU has entered background mode
Halt	$\overline{HALT}$	Suspend external bus activity
Instruction Pipeline	$\overline{IFETCH}$ , $\overline{IPIPE}$	Indicate instruction pipeline activity
Interrupt Request Level	$\overline{IRQ}[7:1]$	Request interrupt service from the CPU
Master In Slave Out	MISO	Serial input to QSPI in master mode; serial output from QSPI in slave mode
Clock Mode Select	MODCLK	Selects system clock source
Master Out Slave In	MOSI	Serial output from QSPI in master mode; serial input to QSPI in slave mode
Port C	PC[6:0]	SIML digital output signals
Peripheral Chip Select	PCS[3:0]	QSPI peripheral chip selects
Port E	PE[7:0]	SIML digital input or output port signals
Port F	PF[7:0]	SIML digital input or output port signals
Port QS	PQS[7:0]	QSM digital I/O port signals
Quotient Out	QUOT	Provides the quotient bit of the polynomial divider
Reset	$\overline{RESET}$	System reset
Read-Modify-Write Cycle	$\overline{RMC}$	Indicates an indivisible read-modify-write instruction
Read/Write	$R/\overline{W}$	Indicates the direction of data transfer on the bus
SCI Receive Data	RXD	Serial input to the SCI
QSPI Serial Clock	SCK	Clock output from QSPI in master mode; clock input to QSPI in slave mode
Size	SIZ[1:0]	Indicates the number of bytes to be transferred during a bus cycle
Slave Select	$\overline{SS}$	Causes serial transmission when QSPI is in slave mode. Causes mode fault in master mode
Three-State Control	TSC	Places all output drivers in a high-impedance state
SCI Transmit Data	TXD	Serial output from the SCI
External Filter Capacitor	XFC	Connection for external phase-locked loop filter capacitor

### 3 Low-Power System Integration Module

The low-power system integration module (SIML) consists of five functional blocks that control system startup, initialization, configuration, and the external bus. **Figure 4** shows the SIML block diagram.



338 S(C)IM BLOCK

**Figure 4 SIML Block Diagram**

#### 3.1 Overview

The system configuration block controls MCU configuration and operating mode.

The clock synthesizer generates clock signals used by the SIML, other IMB modules, and external devices. In addition, a periodic interrupt generator supports execution of time-critical control routines.

The system protection block provides bus and software watchdog monitors.

The chip-select block provides eleven general-purpose chip-select signals and a boot ROM chip-select signal. Both general-purpose and boot ROM chip-select signals have associated base address registers and option registers.

The external bus interface handles the transfer of information between IMB modules and external address space.

The system test block incorporates hardware necessary for testing the MCU. It is used to perform factory tests, and its use in normal applications is not supported.

**Table 7** shows the SIML address map, which occupies 128 bytes. Unused registers within the 128-byte address space return zeros when read. The “Access” column indicates which registers are accessible only at the supervisor privilege level and which can be assigned to either the supervisor or user privilege level, according to the value of the SUPV bit in the SIMLCR.

**Table 7 SIML Address Map**

Access	Address <sup>1</sup>	15	8	7	0
S	\$YFFA00	SIML Module Configuration Register (SIMLCR)			
S	\$YFFA02	SIML Test Register (SIMLTR)			
S	\$YFFA04	Clock Synthesizer Control Register (SYNCR)			
S	\$YFFA06	Not Used		Reset Status Register (RSR)	
S	\$YFFA08	SIML Test Register E (SIMLTRE)			
—	\$YFFA0A	Not Used			
—	\$YFFA0C	Not Used			
—	\$YFFA0E	Not Used			
S/U	\$YFFA10	Not Used		Port E Data (PORTE0)	
S/U	\$YFFA12	Not Used		Port E Data (PORTE1)	
S/U	\$YFFA14	Not Used		Port E Data Direction (DDRE)	
S	\$YFFA16	Not Used		Port E Pin Assignment (PEPAR)	
S/U	\$YFFA18	Not Used		Port F Data (PORTF0)	
S/U	\$YFFA1A	Not Used		Port F Data (PORTF1)	
S/U	\$YFFA1C	Not Used		Port F Data Direction (DDRF)	
S	\$YFFA1E	Not Used		Port F Pin Assignment (PFPAR)	
S	\$YFFA20	Not Used		System Protection Control (SYPCR)	
S	\$YFFA22	Periodic Interrupt Control Register (PICR)			
S	\$YFFA24	Periodic Interrupt Timer Register (PITR)			
S	\$YFFA26	Not Used		Software Service (SWSR)	
S	\$YFFA28	Not Used			
S	\$YFFA2A	Not Used			
S	\$YFFA2C	Not Used			
S	\$YFFA2E	Not Used			
S	\$YFFA30	Test Module Master Shift A (TSTMSRA)			
S	\$YFFA32	Test Module Master Shift B (TSTMSRB)			
S	\$YFFA34	Test Module Shift Count (TSTSC)			
S	\$YFFA36	Test Module Repetition Counter (TSTRC)			
S	\$YFFA38	Test Module Control (CREG)			
S/U	\$YFFA3A	Test Module Distributed Register (DREG)			
—	\$YFFA3C	Not Used			
—	\$YFFA3E	Not Used			
S/U	\$YFFA40	Not Used		Port C Data (PORTC)	
—	\$YFFA42	Not Used			
S	\$YFFA44	Chip-Select Pin Assignment (CSPAR0)			
S	\$YFFA46	Chip-Select Pin Assignment (CSPAR1)			
S	\$YFFA48	Chip-Select Base Boot (CSBARBT)			
S	\$YFFA4A	Chip-Select Option Boot (CSORBT)			
S	\$YFFA4C	Chip-Select Base 0 (CSBAR0)			
S	\$YFFA4E	Chip-Select Option 0 (CSOR0)			
S	\$YFFA50	Chip-Select Base 1 (CSBAR1)			

**Table 7 SIML Address Map (Continued)**

Access	Address <sup>1</sup>	15	8	7	0
S	\$YFFA52				Chip-Select Option 1 (CSOR1)
S	\$YFFA54				Chip-Select Base 2 (CSBAR2)
S	\$YFFA56				Chip-Select Option 2 (CSOR2)
S	\$YFFA58				Chip-Select Base 3 (CSBAR3)
S	\$YFFA5A				Chip-Select Option 3 (CSOR3)
S	\$YFFA5C				Chip-Select Base 4 (CSBAR4)
S	\$YFFA5E				Chip-Select Option 4 (CSOR4)
S	\$YFFA60				Chip-Select Base 5 (CSBAR5)
S	\$YFFA62				Chip-Select Option 5 (CSOR5)
S	\$YFFA64				Chip-Select Base 6 (CSBAR6)
S	\$YFFA66				Chip-Select Option 6 (CSOR6)
S	\$YFFA68				Chip-Select Base 7 (CSBAR7)
S	\$YFFA6A				Chip-Select Option 7 (CSOR7)
S	\$YFFA6C				Chip-Select Base 8 (CSBAR8)
S	\$YFFA6E				Chip-Select Option 8 (CSOR8)
S	\$YFFA70				Chip-Select Base 9 (CSBAR9)
S	\$YFFA72				Chip-Select Option 9 (CSOR9)
S	\$YFFA74				Chip-Select Base 10 (CSBAR10)
S	\$YFFA76				Chip-Select Option 10 (CSOR10)
—	\$YFFA78				Not Used
—	\$YFFA7A				Not Used
—	\$YFFA7C				Not Used
—	\$YFFA7E				Not Used

NOTES:

1. Y = M111, where M is the logic state of the module mapping (MM) bit in the SIMLCR.

### 3.2 System Configuration Block

This functional block provides configuration control for the entire MCU. It also performs interrupt arbitration, bus monitoring, and system test functions.

#### 3.2.1 MCU Configuration

The SIML controls MCU configuration during normal operation and during internal testing.

#### SIMLCR — SIML Configuration Register

**\$YFFA00**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXOFF	FRZSW	FRZBM	0	SLVEN	0	SHEN	SUPV	MM	0	0	IARB[3:0]				

RESET:

0 0 0 0 DATA11 0 0 0 1 1 0 0 1 1 1 1

The SIML configuration register controls system configuration. It can be read or written at any time, except for the module mapping (MM) bit, which can be written only once.

EXOFF — External Clock Off

- 0 = The CLKOUT pin is driven by the MCU system clock.
- 1 = The CLKOUT pin is placed in a high-impedance state.

FRZSW — Freeze Software Enable

- 0 = When FREEZE is asserted, the software watchdog and periodic interrupt timer counters continue to run.
- 1 = When FREEZE is asserted, the software watchdog and periodic interrupt timer counters are disabled, preventing interrupts while the MCU is in background debug mode.

FRZBM — Freeze Bus Monitor Enable

- 0 = When FREEZE is asserted, the bus monitor continues to operate.
- 1 = When FREEZE is asserted, the bus monitor is disabled.

SLVEN — Factory Test Mode Enabled

This bit is a read-only status bit that reflects the state of DATA11 during reset.

- 0 = IMB is not available to an external master.
- 1 = An external bus master has direct access to the IMB.

SHEN[1:0] — Show Cycle Enable

This field determines what the EBI does with the external bus during internal transfer operations. A show cycle allows internal transfers to be externally monitored. **Table 8** shows whether show cycle data is driven externally, and whether external bus arbitration can occur. To prevent bus conflict, external peripherals must not be enabled during show cycles.

**Table 8 Show Cycle Enable Bits**

SHEN	Action
00	Show cycles disabled, external bus arbitration allowed
01	Show cycles enabled, external bus arbitration not allowed
10	Show cycles enabled, external bus arbitration allowed
11	Show cycles enabled, external bus arbitration allowed, internal activity is halted by a bus grant

SUPV — Supervisor/Unrestricted Data Space

The SUPV bit places the SIML global registers in either supervisor or user data space.

- 0 = Registers with access controlled by the SUPV bit are accessible in either supervisor or user data space.
- 1 = Registers with access controlled by the SUPV bit are accessible in supervisor data space only.

MM — Module Mapping

- 0 = Internal modules are addressed from \$7FF000 – \$7FFFFFF.
- 1 = Internal modules are addressed from \$FFF000 – \$FFFFFF.

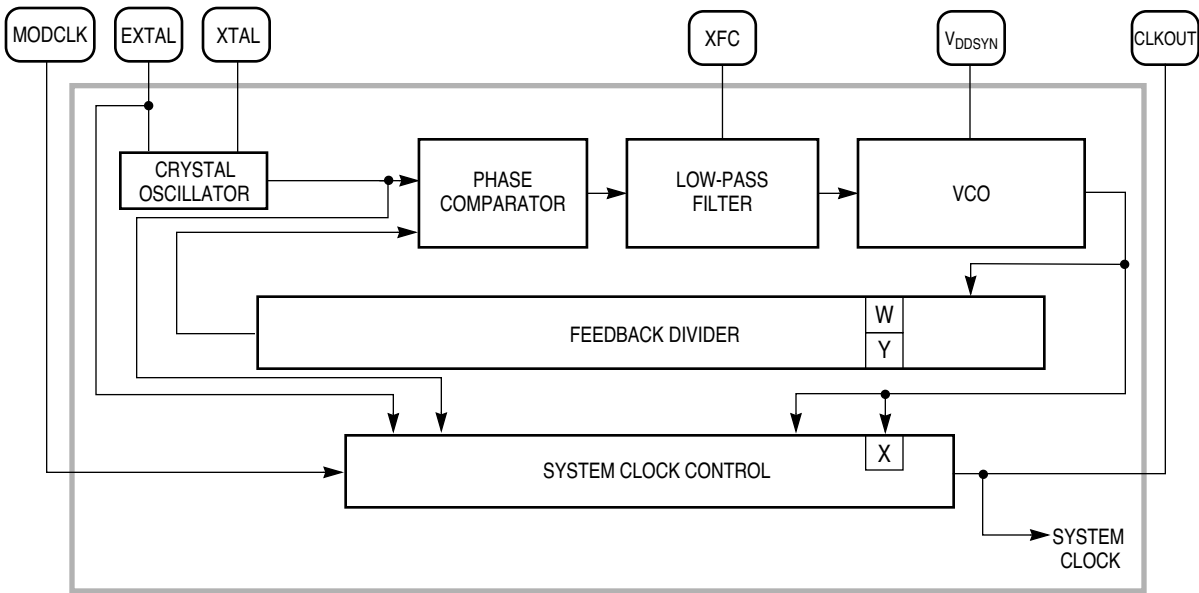
IARB[3:0] — Interrupt Arbitration Field

Each module that can generate interrupt requests has an interrupt arbitration (IARB) field. Arbitration between interrupt requests of the same priority is performed by serial contention between IARB field bit values. Contention must take place whenever an interrupt request is acknowledged, even when there is only a single pending request. An IARB field must have a non-zero value for contention to take place. If an interrupt request from a module with an IARB field value of %0000 is recognized, the CPU processes a spurious interrupt exception. Because the SIML routes external interrupt requests to the CPU, the SIML IARB field value is used for arbitration between internal and external interrupts of the same priority. The reset value of IARB for the SIML is %1111, and the reset value of IARB for all other modules is %0000, which prevents SIML interrupts from being discarded during initialization.

### 3.3 System Clock

The system clock in the SIML provides timing signals for the IMB modules and for an external peripheral bus. Because the MCU is a fully static design, register and memory contents are not affected when the clock rate changes. System hardware and software support changes in clock rate during operation.

The system clock signal can be generated in one of two ways. An internal phase-locked loop can synthesize the clock from a reference frequency, or the clock signal can be input directly from an external source. Keep these clock sources in mind while reading the rest of this section. **Figure 5** is a block diagram of the system clock.



16/32 PLL BLOCK

**Figure 5 System Clock Block Diagram**

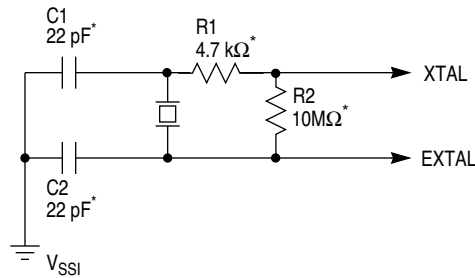
#### 3.3.1 Clock Sources

The state of the clock mode (MODCLK) pin during reset determines the system clock source. When MODCLK is held high during reset, the clock synthesizer generates a clock signal from a reference frequency connected to the EXTAL pin. The clock synthesizer control register (SYNCR) determines operating frequency and mode of operation. When MODCLK is held low during reset, the clock synthesizer is disabled and an external system clock signal must be applied. The SYNCR control bits have no effect.

The input clock is referred to as " $f_{ref}$ ", and can be either a crystal or an external clock source. The output of the clock system is referred to as " $f_{sys}$ ". Ensure that  $f_{ref}$  and  $f_{sys}$  are within normal operating limits.

The reference frequency for this MCU is typically 32.768 kHz, but can range from 25 kHz to 50 kHz. To generate a reference frequency using the crystal oscillator, a reference crystal must be connected between the EXTAL and XTAL pins. **Figure 6** shows a recommended circuit.





\* RESISTANCE AND CAPACITANCE BASED ON A TEST CIRCUIT CONSTRUCTED WITH A DAISHINKU DMX-38 32.768 KHZ CRYSTAL. SPECIFIC COMPONENTS MUST BE BASED ON CRYSTAL TYPE. CONTACT CRYSTAL VENDOR FOR EXACT CIRCUIT.

338 OSCILLATOR

**Figure 6 System Clock Oscillator Circuit**

When an external system clock signal is applied (PLL disabled, MODCLK = 0 during reset), the duty cycle of the input is critical, especially at operating frequencies close to maximum. The relationship between clock signal duty cycle and clock signal period is expressed:

$$\text{Minimum External Clock Period} = \frac{\text{Minimum External Clock High/Low Time}}{50\% - \text{Percentage Variation of External Clock Input Duty Cycle}}$$

When the system clock signal is applied directly to the EXTAL pin (PLL is disabled, MODCLK = 0 during reset), or the clock synthesizer reference frequency is supplied by a source other than a crystal (PLL enabled, MODCLK = 1 during reset), the XTAL pin must be left floating. In either case, the frequency of the signal applied to EXTAL may not exceed the maximum system clock frequency (PLL disabled) or the maximum clock synthesizer reference frequency (PLL enabled).

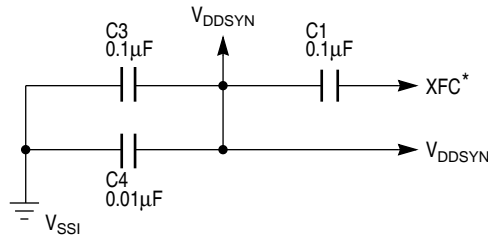
### 3.3.2 Clock Synthesizer Operation

$V_{DDSYN}$  is used to power the clock circuits when the phase-locked loop is used. A separate power source increases MCU noise immunity and can be used to run the clock when the MCU is powered down. A quiet power supply must be used as the  $V_{DDSYN}$  source. Adequate external bypass capacitors should be placed as close as possible to the  $V_{DDSYN}$  pin to assure stable operating frequency. When an external system clock signal is applied and the PLL is disabled,  $V_{DDSYN}$  should be connected to the  $V_{DD}$  supply. Refer to the *SIM Reference Manual (SIMRM/AD)* for more information regarding system clock power supply conditioning.

A voltage controlled oscillator (VCO) generates the system clock signal. To maintain a 50% clock duty cycle, the VCO frequency ( $f_{VCO}$ ) is either two or four times the system clock frequency, depending on the state of the X bit in SYNCR. A portion of the clock signal is fed back to a divider/counter. The divider controls the frequency of one input to a phase comparator. The other phase comparator input is the reference signal connected to the EXTAL pin. The comparator generates a control signal proportional to the difference in phase between the two inputs. The signal is low-pass filtered and used to correct the VCO output frequency.

Filter circuit implementation can vary, depending upon the external environment and required clock stability. **Figure 7** shows a recommended system clock filter network. XFC pin leakage must be kept within specified limits to maintain optimum stability and PLL performance.

An external filter network connected to the XFC pin is not required when an external system clock signal is applied and the PLL is disabled. The XFC pin must be left floating in this case.



\* MAINTAIN LOW LEAKAGE ON THE XFC NODE.

32 XFC CONN

**Figure 7 System Clock Filter Network**

When the clock synthesizer is used, SYNCR determines the operating frequency of the MCU. The following equation relates the MCU operating frequency to the clock synthesizer reference frequency ( $f_{ref}$ ) and the W, X, and Y fields in the SYNCR:

$$f_{sys} = 4f_{ref}(Y + 1)(2^{2W + X})$$

The W bit controls a prescaler tap in the feedback divider. Setting W increases VCO speed by a factor of four. The Y field determines the count modulus for a modulo 64 downcounter, causing it to divide by a value of Y+1. When W or Y changes, VCO frequency ( $f_{VCO}$ ) changes, and the VCO must relock.

The X bit controls a divide-by-two circuit that is not in the synthesizer feedback loop. When X=0 (reset state), the divider is enabled, and the system clock is one-fourth the VCO frequency. Setting X=1 disables the divider, doubling the clock speed without changing the VCO frequency. There is no relock delay when clock speed is changed by the X bit.

Internal VCO frequency is determined by the following equations:

$$f_{VCO} = 4f_{sys} \text{ if } X = 0$$

or

$$f_{VCO} = 2f_{sys} \text{ if } X = 1$$

For the MCU to operate correctly, system clock and VCO frequencies selected by the W, X, and Y bits must be within the limits specified for the MCU. Do not use a combination of bit values that selects either an operating frequency or a VCO frequency greater than the maximum specified values.

### 3.3.3 Clock Synthesizer Control

The clock synthesizer control circuits determine system clock frequency and clock operation under special circumstances, such as following loss of synthesizer reference or during low-power operation. Clock source is determined by the logic state of the MODCLK pin during reset.

#### SYNCR — Clock Synthesizer Control Register

**\$YFFA04**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	X	Y						EDIV	STCPU	0	RSVD <sup>1</sup>	SLOCK	RSVD <sup>1</sup>	STSIM	STEXT

RESET:

0 0 1 1 1 1 1 1 0 0 0 0 U 0 0 0

NOTES:

1. Ensure that initialization software does not change the value of this bit (it should always be zero).

When the on-chip clock synthesizer is used, system clock frequency is controlled by the bits in the upper byte of SYNCR. Bits in the lower byte show the status of or control the operation of internal and external clocks. SYNCR can be read or written only when the CPU is operating in supervisor mode.

**W** — Frequency Control (VCO)

This bit controls a prescaler tap in the synthesizer feedback loop. Setting it increases the VCO speed by a factor of four. VCO relock delay is required.

**X** — Frequency Control (Prescaler)

This bit controls a divide by two prescaler that is not in the synthesizer feedback loop. Setting it doubles the clock speed without changing the VCO speed. No VCO relock delay is required.

**Y[5:0]** — Frequency Control (Counter)

The Y field controls the modulus down counter in the synthesizer feedback loop, causing it to divide by a value of Y + 1. Values range from 0 to 63. VCO relock delay is required.

**EDIV** — E Clock Divide Rate

0 = ECLK frequency is system clock divided by 8.

1 = ECLK frequency is system clock divided by 16.

ECLK is an external M6800 bus clock available on pin ADDR23. Refer to **3.6 Chip-Selects** for more information.

**STCPU** — Stop CPU32L Clock on LPSTOP

0 = When LPSTOP is executed, the intermodule bus clock (IMBCLK) is held low. When a trace, reset exception, or SIML interrupt occurs, the IMBCLK turns back on and the CPU32L begins executing instructions again.

1 = When LPSTOP is executed, the IMBCLK continues to run but is gated off and held low only where it enters the CPU32L. When a trace, reset exception, or interrupt from any module occurs, the IMBCLK is gated back on where it enters the CPU32L, and execution begins again.

**SLOCK** — Synthesizer Lock Flag

0 = VCO has not locked, but is enabled on the desired frequency.

1 = VCO has locked on the desired frequency, or is disabled.

The MCU remains in reset until the synthesizer locks, but SLOCK does not indicate synthesizer lock status until after the user writes to SYNCR.

**STSIM** — Stop Mode SIML Clock

0 = When LPSTOP is executed, the SIML clock is driven by the crystal oscillator and the VCO is turned off to conserve power.

1 = When LPSTOP is executed, the SIML clock is driven by the VCO.

**STEXT** — Stop Mode External Clock

0 = When LPSTOP is executed, the CLKOUT signal is held negated to conserve power.

1 = When LPSTOP is executed, the CLKOUT signal is driven by the SIML clock, as determined by the state of the STSIM bit.

### 3.3.4 External MC6800 Bus Clock

The state of the ECLK division rate bit (EDIV) in SYNCR determines clock rate for the ECLK signal available on pin ADDR23. ECLK is a bus clock for MC6800 devices and peripherals. ECLK frequency can be set to system clock frequency divided by eight or system clock frequency divided by sixteen. The clock is enabled by the  $\overline{CS10}$  field in chip-select pin assignment register 1 (CSPAR1). ECLK operation during low-power stop is described in the following paragraph. Refer to **3.6 Chip-Selects** for more information about the external bus clock.

### 3.3.5 Low-Power Operation

Low-power operation is initiated by the CPU32L. To reduce power consumption selectively, the CPU32L can enter the following low-power modes:

1. The CPU32L can selectively disable a module by setting the module's STOP bit.
2. The CPU32L can execute the STOP instruction.
3. The CPU32L can execute the LPSTOP instruction to stop the operations of only the CPU32L or the entire MCU, including the CPU32L.

If the STOP bit in a module is set, then that module enters a low power mode. Some or all of that module's registers remain accessible. The module can be restarted by asserting  $\overline{\text{RESET}}$  or by the CPU32L clearing the module's STOP bit.

The CPU32L can enter a low power mode by executing the STOP instruction. It can be reawakened by  $\overline{\text{RESET}}$ , trace or interrupt.

#### 3.3.5.1 Normal LPSTOP mode

This low-power stop mode offers the greatest power reduction. To enter normal LPSTOP mode, the CPU32L executes the LPSTOP instruction after clearing the STCPU bit in SYNCR. This causes the SIML to turn off the system clock to most of the MCU.

When the CPU executes LPSTOP, a special CPU space bus cycle writes a copy of the current interrupt mask into the clock control logic. The SIML brings the MCU out of normal LPSTOP mode when one of the following exceptions occurs:

- $\overline{\text{RESET}}$
- Trace
- SIML interrupt of higher priority than the stored interrupt mask

During a LPSTOP, unless the system clock signal is supplied by an external source and that source is removed, the SIML clock control logic and the SIML clock signal (SIMCLK) continue to operate. The periodic interrupt timer and input logic for the  $\overline{\text{RESET}}$  and  $\overline{\text{IRQ}}$  pins are clocked by SIMCLK, and can be used to bring the processor out of LPSTOP. The software watchdog monitor cannot perform this function. Optionally, the SIML can also continue to generate the CLKOUT signal while in LPSTOP.

STSIM and STEXT bits in SYNCR determine clock operation during LPSTOP.

#### 3.3.5.2 Modified LPSTOP mode

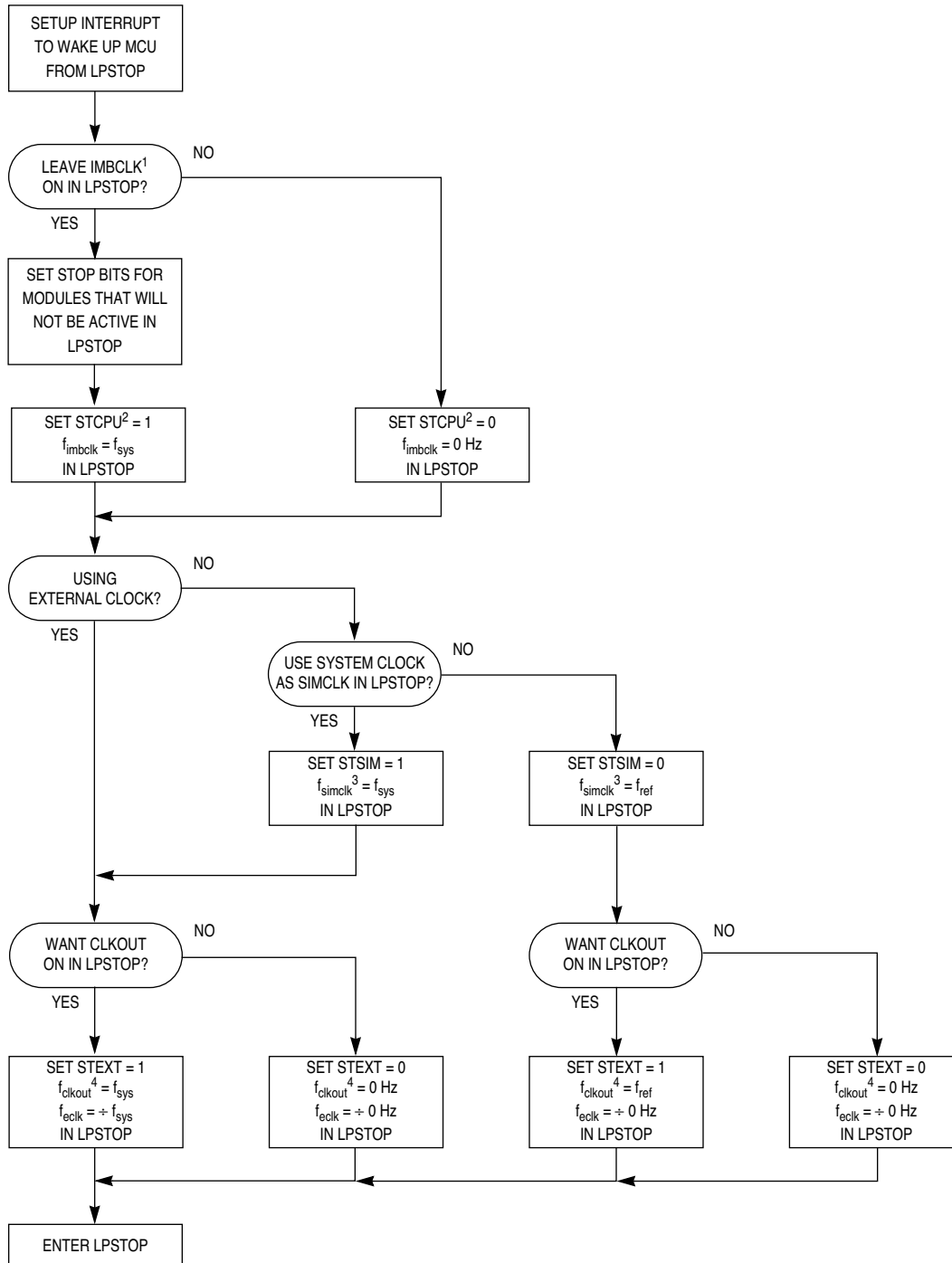
To enter modified LPSTOP mode, the CPU32L first sets the STCPU bit in SYNCR, then executes the LPSTOP instruction. This causes the SIML to turn off the system clock to the CPU32L only. The other MCU modules continue to operate. The SIML brings the MCU out of normal LPSTOP mode when one of the following exceptions occurs:

- $\overline{\text{RESET}}$
- Trace
- Interrupt of higher priority than the stored interrupt mask from any MCU module

This low-power stop mode offers better power reduction than using the STOP instruction since the clock in the CPU32L is held inactive. Also, the STOP bits of individual modules may be set or cleared, leaving some active and others inactive. The flow chart shown in **Figure 8** summarizes the effects of the STCPU, STSIM, and STEXT bits when the MCU enters normal or modified LPSTOP mode.

#### NOTE

To keep power consumption to a minimum when in LPSTOP mode, do not allow any spurious interrupts to occur. If a spurious interrupt occurs during LPSTOP mode, the device will transition to the STOP mode (which has greater power consumption) until a non-spurious interrupt request is detected by the CPU32L.



NOTES:

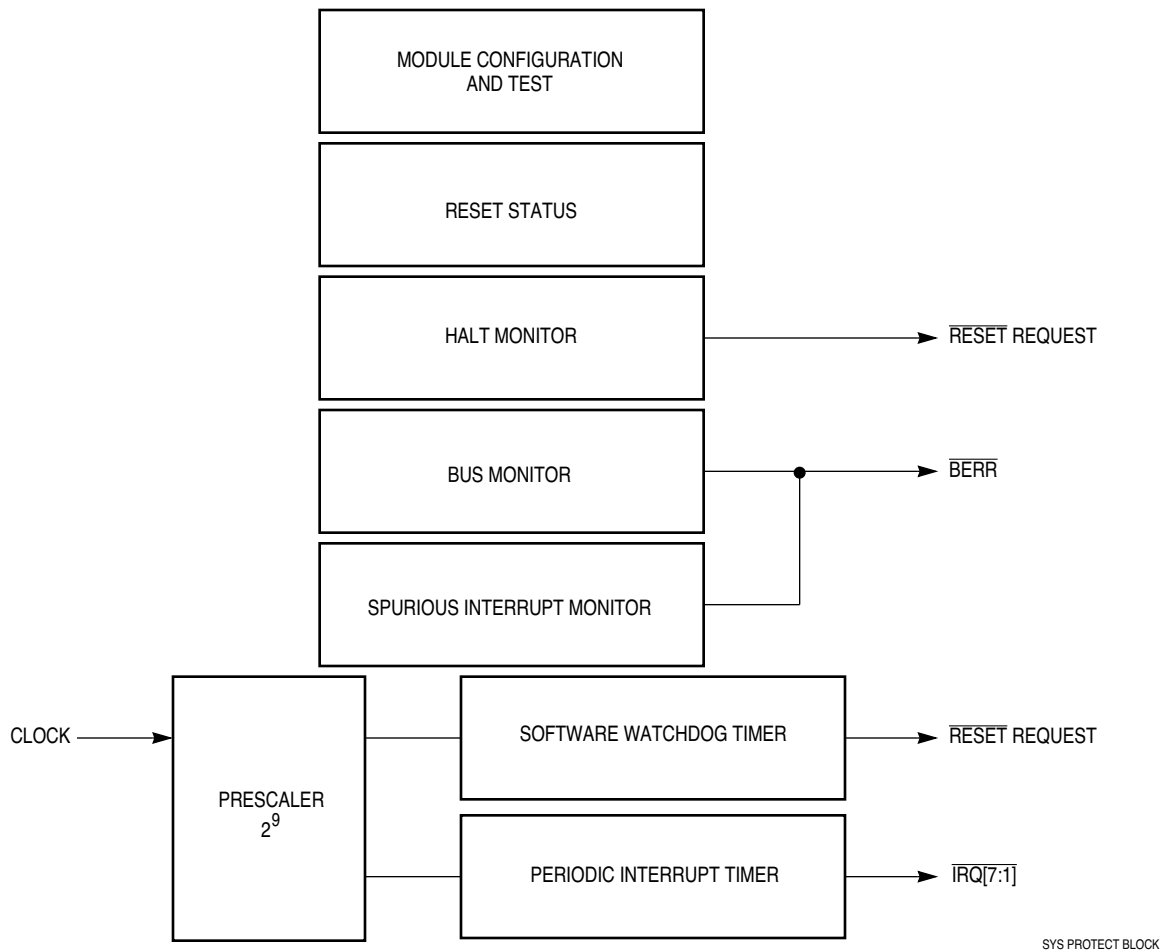
1. IMBCLK IS THE CLOCK USED BY THE CPU32L, QSM, CTM6, AND THE SIML.
2. WHEN STCPU = 1, THE CPU32L IS SHUTDOWN IN LPSTOP. ALL OTHER MODULES WILL REMAIN ACTIVE UNLESS THE STOP BITS IN THEIR MODULE CONFIGURATION REGISTERS ARE SET PRIOR TO ENTERING LPSTOP.
3. THE SIMCLK IS USED BY THE PIT, IRQ, AND INPUT BLOCKS OF THE SIML.
4. CLKOUT CONTROL DURING LPSTOP IS OVERRIDDEN BY THE EXOFF BIT IN SIMLCR. IF EXOFF = 1, THE CLKOUT PIN IS ALWAYS IN A HIGH IMPEDANCE STATE AND STEXT HAS NO EFFECT IN LPSTOP. IF EXOFF = 0, CLKOUT IS CONTROLLED BY STEXT IN LPSTOP. WHEN STCPU = 1, THE CPU32L IS DISABLED IN LPSTOP, BUT ALL OTHER MODULES REMAIN ACTIVE OR STOPPED ACCORDING TO THE SETTING.

LPSTOP FLOWCHART

Figure 8 LPSTOP Flowchart

### 3.4 System Protection Block

System protection includes a bus monitor, a halt monitor, a spurious interrupt monitor, and a software watchdog timer. These functions reduce the number of external components required for complete system control. **Figure 9** shows the system protection block.



**Figure 9 System Protection Block**

#### 3.4.1 System Protection Control Register

The system protection control register controls the software watchdog timer, bus monitor, and halt monitor. This register can be written only once following power-on or reset, but can be read at any time.

**SYPCCR** — System Protection Control Register

**\$YFFA21**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
NOT USED								SWE	SWP	SWT[1:0]		HME	BME	BMT		
RESET:								1	MODCLK	0	0	0	0	0	0	0

**SWE** — Software Watchdog Enable  
 0 = Software watchdog disabled  
 1 = Software watchdog enabled

### SWP — Software Watchdog Prescaler

This bit controls the value of the software watchdog prescaler.

0 = Software watchdog clock not prescaled

1 = Software watchdog clock prescaled by 512

### SWT[1:0] — Software Watchdog Timing

This field selects the divide ratio used to establish software watchdog time-out period. **Table 9** gives the ratio for each combination of SWP and SWT bits.

**Table 9 Software Watchdog Timing Field**

SWP	SWT[1:0]	Watchdog Time-Out Period
0	00	$2^9 \div f_{\text{sys}}$
0	01	$2^{11} \div f_{\text{sys}}$
0	10	$2^{13} \div f_{\text{sys}}$
0	11	$2^{15} \div f_{\text{sys}}$
1	00	$2^{18} \div f_{\text{sys}}$
1	01	$2^{20} \div f_{\text{sys}}$
1	10	$2^{22} \div f_{\text{sys}}$
1	11	$2^{24} \div f_{\text{sys}}$

### HME — Halt Monitor Enable

0 = Disable halt monitor function

1 = Enable halt monitor function

### BME — Bus Monitor Enable

0 = Disable bus monitor function for internal to external bus cycles.

1 = Enable bus monitor function for internal to external bus cycles.

### BMT[1:0] — Bus Monitor Timing

This bit field selects the time-out period in system clocks for the bus monitor. Refer to **Table 10**.

**Table 10 Bus Monitor Time-Out Period**

BMT[1:0]	Bus Monitor Time-Out Period
00	64 system clocks
01	32 system clocks
10	16 system clocks
11	8 system clocks

## 3.4.2 Bus Monitor

The internal bus monitor checks for excessively long  $\overline{\text{DSACK}}$  response times during normal bus cycles and for excessively long  $\overline{\text{DSACK}}$  or  $\overline{\text{AVEC}}$  response times during interrupt acknowledge (IACK) cycles. The monitor asserts  $\overline{\text{BERR}}$  if the response time exceeds a user-specified time-out period.

$\overline{\text{DSACK}}$  and  $\overline{\text{AVEC}}$  response times are measured in clock cycles. The maximum allowable response time can be selected by setting the BMT[1:0] field.

The monitor does not check  $\overline{DSACK}$  response on the external bus unless the CPU initiates the bus cycle. The BME bit in SYPCR enables the internal bus monitor for internal to external bus cycles. If a system contains external bus masters, an external bus monitor must be implemented and the internal to external bus monitor option must be disabled.

### 3.4.3 Halt Monitor

The halt monitor responds to assertion of the  $\overline{HALT}$  signal on the internal bus caused by a double bus fault. A double bus fault occurs when:

- Bus error exception processing begins and a second  $\overline{BERR}$  is detected before the first instruction of the first exception handler is executed.
- One or more bus errors occur before the first instruction after a reset exception is executed.
- A bus error occurs while the CPU is loading information from a bus error stack frame during a return from exception (RTE) instruction.

If the halt monitor is enabled by setting HME in SYPCR, the MCU will issue a reset when a double bus fault occurs, otherwise the MCU will remain halted.

A flag in the reset status register (RSR) indicates that the last reset was caused by the halt monitor.

### 3.4.4 Spurious Interrupt Monitor

The spurious interrupt monitor issues  $\overline{BERR}$  if no interrupt arbitration occurs during an interrupt acknowledge cycle. Leaving IARB[3:0] set to %0000 in the module configuration register of any peripheral that can generate interrupts will cause a spurious interrupt.

### 3.4.5 Software Watchdog

The software watchdog is controlled by SWE in SYPCR. Once enabled, the watchdog requires that a service sequence be written to SWSR on a periodic basis. If servicing does not take place, the watchdog times out and issues a reset. This register can be written at any time, but returns zeros when read.

#### SWSR — Software Service Register \$YFFA27

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
NOT USED								SWSR								
RESET:																
								0	0	0	0	0	0	0	0	0

Each time the service sequence is written, the software watchdog timer restarts. The servicing sequence consists of the following steps:

1. Write \$55 to SWSR.
2. Write \$AA to SWSR.

Both writes must occur before time-out in the order listed, but any number of instructions can be executed between the two writes.

The watchdog clock rate is affected by SWP and SWT[1:0] in SYPCR. When SWT[1:0] are modified, a watchdog service sequence must be performed before the new time-out period takes effect.

The reset value of SWP is affected by the state of the MODCLK pin on the rising edge of  $\overline{RESET}$ . Refer to **Table 11**.



**Table 11 MODCLK Pin States**

MODCLK	SWP
0	1
1	0

### 3.4.6 Periodic Interrupt Timer

The periodic interrupt timer (PIT) generates interrupts at user-programmable intervals. Timing for the PIT is provided by a programmable prescaler driven by the system clock.

#### PICR — Periodic Interrupt Control Register

**\$YFFA22**

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

This register contains information concerning periodic interrupt priority and vectoring. Bits [10:0] can be read or written at any time. Bits [15:11] are unimplemented and always return zero.

PIRQL[2:0] — Periodic Interrupt Request Level

**Table 12** shows what interrupt request level is asserted when a periodic interrupt is generated. If a PIT interrupt and an external  $\overline{IRQ}$  signal of the same priority occur simultaneously, the PIT interrupt is serviced first. The periodic timer continues to run when the interrupt is disabled.

**Table 12 Periodic Interrupt Request Levels**

PIRQL[2:0]	Interrupt Request Level
000	Periodic Interrupt Disabled
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

#### PIV[7:0] — Periodic Interrupt Vector

This bit field contains the vector generated in response to an interrupt from the periodic timer. When the SIML responds, the periodic interrupt vector is placed on the bus.

#### PITR — Periodic Interrupt Timer Register

**\$YFFA24**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	PTP	PITM[7:0]							
RESET:															
0	0	0	0	0	0	0	MODCLK	0	0	0	0	0	0	0	0

PITR contains the count value for the periodic timer. Setting the PITM[7:0] field turns off the periodic timer. This register can be read or written at any time.

PTP — Periodic Timer Prescaler Control

0 = Periodic timer clock not prescaled

1 = Periodic timer clock prescaled by 512

The reset state of PTP is the complement of the state of the MODCLK signal at the rising edge of  $\overline{\text{RESET}}$ .

PITM[7:0] — Periodic Interrupt Timer Modulus

This is an 8-bit timing modulus. The period of the timer can be calculated as follows:

$$\text{PIT Period} = \frac{4(\text{PITM}[7:0])(\text{Prescaler})}{f_{\text{ref}}}$$

where

PIT Period = Periodic interrupt timer period

PITM[7:0] = Periodic interrupt timer register modulus

$f_{\text{ref}}$  = Synthesizer reference of external clock input frequency

Prescaler = 1 if PTP = 0 or 512 if PTP = 1

### 3.5 External Bus Interface

The external bus interface (EBI) transfers information between the internal MCU bus and external devices. The external bus has 24 address lines and 16 data lines.

The EBI provides dynamic sizing between 8-bit and 16-bit data accesses. It supports byte, word, and long-word transfers. Ports are accessed through the use of asynchronous cycles controlled by the size (SIZ1 and SIZ0) and data size acknowledge ( $\overline{\text{DSACK1}}$  and  $\overline{\text{DSACK0}}$ ) pins. Multiple bus cycles may be required for dynamically sized transfer.

Port width is the maximum number of bits accepted or provided during a bus transfer. External devices must follow the handshake protocol described below. Control signals indicate the beginning of the cycle, the address space, the size of the transfer, and the type of cycle. The selected device controls the length of the cycle. Strobe signals, one for the address bus and another for the data bus, indicate the validity of an address and provide timing information for data. The EBI operates in an asynchronous mode for any port width.

To add flexibility and minimize the necessity for external logic, MCU chip-select logic can be synchronized with EBI transfers. Chip-select logic can also provide internally-generated bus control signals for these accesses. Refer to **3.6 Chip-Selects** for more information.

#### 3.5.1 Bus Control Signals

The CPU initiates a bus cycle by driving the address, size, function code, and read/write outputs. At the beginning of the cycle, size signals SIZ0 and SIZ1 are driven along with the function code signals (FC[2:0]). The size signals indicate the number of bytes remaining to be transferred during an operand cycle. They are valid while the address strobe  $\overline{\text{AS}}$  is asserted.

**Table 13** shows SIZ0 and SIZ1 encoding. The read/write ( $\overline{\text{R/W}}$ ) signal determines the direction of the transfer during a bus cycle. This signal changes state, when required, at the beginning of a bus cycle, and is valid while  $\overline{\text{AS}}$  is asserted. The  $\overline{\text{R/W}}$  signal only changes state when a write cycle is preceded by a read cycle or vice versa. The signal can remain low for two consecutive write cycles.

**Table 13 Size Signal Encoding**

SIZ1	SIZ0	Transfer Size
0	1	Byte
1	0	Word
1	1	Three Byte
0	0	Long Word

### 3.5.2 Function Codes

The CPU32L automatically generates function code signals FC[2:0]. The function codes can be considered address extensions that automatically select one of eight address spaces to which an address applies. These spaces are designated as either user or supervisor, and program or data spaces. Address space seven is designated CPU space. CPU space is used for control information not normally associated with read or write bus cycles. Function codes are valid while  $\overline{AS}$  is asserted. **Table 14** displays CPU32L address space encodings.

**Table 14 CPU32L Address Space Encoding**

FC2	FC1	FC0	Address Space
0	0	0	Reserved
0	0	1	User Data Space
0	1	0	User Program Space
0	1	1	Reserved
1	0	0	Reserved
1	0	1	Supervisor Data Space
1	1	0	Supervisor Program Space
1	1	1	CPU Space

### 3.5.3 Address Bus

Address bus signals ADDR[23:0] define the address of the most significant byte to be transferred during a bus cycle. The MCU places the address on the bus at the beginning of a bus cycle. The address is valid while  $\overline{AS}$  is asserted.

### 3.5.4 Address Strobe

$\overline{AS}$  is a timing signal that indicates the validity of an address on the address bus and the validity of many control signals. It is asserted one-half clock after the beginning of a bus cycle.

### 3.5.5 Data Bus

Data bus signals DATA[15:0] make up a bidirectional, non-multiplexed parallel bus that transfers data to or from the MCU. A read or write operation can transfer 8 or 16 bits of data in one bus cycle. During a read cycle, the data is latched by the MCU on the last falling edge of the clock for that bus cycle. For a write cycle, all 16 bits of the data bus are driven, regardless of the port width or operand size. The MCU places the data on the data bus one-half clock cycle after  $\overline{AS}$  is asserted in a write cycle.

### 3.5.6 Data Strobe

Data strobe ( $\overline{DS}$ ) is a timing signal. For a read cycle, the MCU asserts  $\overline{DS}$  to signal an external device to place data on the bus.  $\overline{DS}$  is asserted at the same time as  $\overline{AS}$  during a read cycle. For a write cycle,  $\overline{DS}$  signals an external device that data on the bus is valid. The MCU asserts  $\overline{DS}$  one full clock cycle after the assertion of  $\overline{AS}$  during a write cycle.

### 3.5.7 Bus Cycle Termination Signals

During bus cycles, external devices assert the data size acknowledge signals  $\overline{DSACK1}$  and  $\overline{DSACK0}$ . During a read cycle, the signals tell the MCU to terminate the bus cycle and to latch data. During a write cycle, the signals indicate that an external device has successfully stored data and that the cycle can end. These signals also indicate to the MCU the size of the port for the bus cycle just completed. Alternatively, chip-selects can be used to generate  $\overline{DSACK1}$  and  $\overline{DSACK0}$  internally. Refer to **3.5.8 Dynamic Bus Sizing** for more information.

The bus error ( $\overline{BERR}$ ) signal is also a bus cycle termination indicator and can be used in the absence of  $\overline{DSACK1}$  and  $\overline{DSACK0}$  to indicate a bus error condition. It can also be asserted in conjunction with these signals, provided it meets the appropriate timing requirements. The internal bus monitor can be used to generate the  $\overline{BERR}$  signal for internal-to-external transfers. When  $\overline{BERR}$  and  $\overline{HALT}$  are asserted simultaneously, the CPU takes a bus error exception.

The autovector signal ( $\overline{AVEC}$ ) can terminate  $\overline{IRQ}$  pin interrupt acknowledge cycles.  $\overline{AVEC}$  indicates that the MCU will internally generate a vector number to locate an interrupt handler routine. If it is continuously asserted, autovectors will be generated for all external interrupt requests.  $\overline{AVEC}$  is ignored during all other bus cycles.

### 3.5.8 Dynamic Bus Sizing

The MCU dynamically interprets the port size of the addressed device during each bus cycle, allowing operand transfers to or from 8- and 16-bit ports. During an operand transfer cycle, the slave device signals its port size and indicates completion of the bus cycle to the MCU through the use of the  $\overline{DSACK1}$  and  $\overline{DSACK0}$  inputs, as shown in **Table 15**.

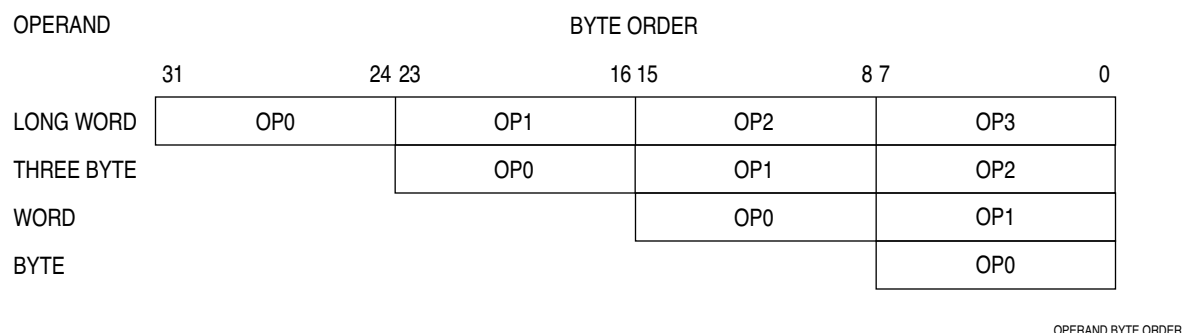
**Table 15 Effect of  $\overline{DSACK}$  Signals**

$\overline{DSACK1}$	$\overline{DSACK0}$	Result
1	1	Insert Wait States in Current Bus Cycle
1	0	Complete Cycle — Data Bus Port Size is 8 Bits
0	1	Complete Cycle — Data Bus Port Size is 16 Bits
0	0	Reserved

For example, if the MCU is executing an instruction that reads a long-word operand from a 16-bit port, the MCU latches the 16 bits of valid data and then runs another bus cycle to obtain the other 16 bits. The operation for an 8-bit port is similar, but requires four read cycles. The addressed device uses the  $\overline{DSACK0}$  and  $\overline{DSACK1}$  signals to indicate the port width. For instance, a 16-bit device always returns  $\overline{DSACK0} = 1$  and  $\overline{DSACK1} = 0$  for a 16-bit port, regardless of whether the bus cycle is a byte or word operation.

Dynamic bus sizing requires that the portion of the data bus used for a transfer to or from a particular port size be fixed. A 16-bit port must reside on data bus bits [15:0] and an 8-bit port must reside on data bus bits [15:8]. This minimizes the number of bus cycles needed to transfer data and ensures that the MCU transfers valid data.

The MCU always attempts to transfer the maximum amount of data on all bus cycles. For a word operation, it is assumed that the port is 16 bits wide when the bus cycle begins. Operand bytes are designated as shown in **Figure 10**. OP0 is the most significant byte of a long-word operand, and OP3 is the least significant byte. The two bytes of a word-length operand are OP0 (most significant) and OP1. The single byte of a byte-length operand is OP0.



**Figure 10 Operand Byte Order**

### 3.5.9 Operand Alignment

The data multiplexer establishes the necessary connections for different combinations of address and data sizes. The multiplexer takes the two bytes of the 16-bit bus and routes them to their required positions. Positioning of bytes is determined by the size and address outputs. SIZ1 and SIZ0 indicate the remaining number of bytes to be transferred during the current bus cycle. The number of bytes transferred is equal to or less than the size indicated by SIZ1 and SIZ0, depending on port width.

ADDR0 also affects the operation of the data multiplexer. During an operand transfer, ADDR[23:1] indicate the word base address of the portion of the operand to be accessed, and ADDR0 indicates the byte offset from the base.

### 3.5.10 Misaligned Operands

CPU32L processor architecture uses a basic operand size of 16 bits. An operand is misaligned when it overlaps a word boundary. This is determined by the value of ADDR0. When ADDR0 = 0 (an even address), the address is on a word and byte boundary. When ADDR0 = 1 (an odd address), the address is on a byte boundary only.

A byte operand is aligned at any address; a word or long-word operand is misaligned at an odd address. The CPU32L does not support misaligned operand transfers, and gives an address error exception if one is attempted.

The largest amount of data that can be transferred by a single bus cycle is an aligned word. If the MCU transfers a long-word operand via a 16-bit port, the most significant operand word is transferred on the first bus cycle and the least significant operand word on a following bus cycle.

### 3.5.11 Operand Transfer Cases

**Table 16** summarizes how operands are aligned for various types of transfers. OPn entries are portions of a requested operand that are read or written during a bus cycle and are defined by SIZ1, SIZ0, and ADDR0 for that bus cycle.

**Table 16 Operand Alignment**

Current Cycle	Transfer Case <sup>1</sup>	SIZ1	SIZ0	ADDR0	$\overline{DSACK1}$	$\overline{DSACK0}$	DATA [15:8]	DATA [7:0]	Next Cycle
1	Byte to 8-bit port (even)	0	1	0	1	0	OP0	(OP0) <sup>2</sup>	—
2	Byte to 8-bit port (odd)	0	1	1	1	0	OP0	(OP0)	—
3	Byte to 16-bit port (even)	0	1	0	0	1	OP0	(OP0)	—
4	Byte to 16-bit port (odd)	0	1	1	0	1	(OP0)	OP0	—
5	Word to 8-bit port	1	0	0	1	0	OP0	(OP1)	2
6	Word to 16-bit port	1	0	0	0	1	OP0	OP1	—
7	3-byte to 8-bit port <sup>3</sup>	1	1	1	1	0	OP0	(OP0)	5
8	Long word to 8-bit port	0	0	0	1	0	OP0	(OP0)	7
9	Long word to 16-bit port	0	0	0	0	1	OP0	OP1	6

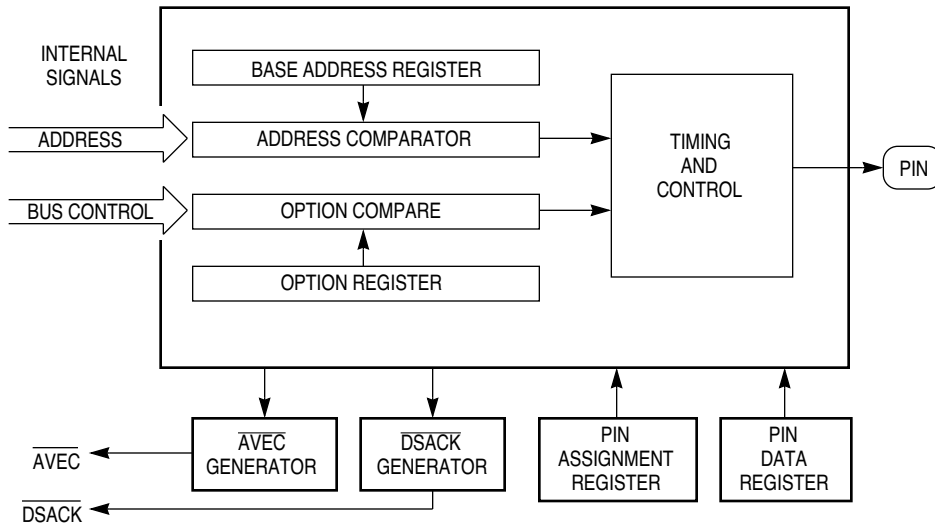
NOTES:

1. All transfers are aligned. The CPU32L does not support misaligned word or long-word transfers.
2. Operands in parentheses are ignored by the CPU32L during read cycles.
3. 3-Byte transfer cases occur only as a result of a long word to 8-bit port transfer.

### 3.6 Chip-Selects

Typical microcontrollers require additional hardware to provide external chip-select and address decode signals. The MC68338 includes 12 programmable chip-selects that can provide 2 to 16-clock-cycle access to external memory and peripherals. Address block sizes of two Kbytes to one Mbyte can be selected. **Figure 11** is a functional diagram of a chip-select circuit.

Chip-select assertion can be synchronized with bus control signals to provide output enable, read/write strobe, or interrupt acknowledge signals. Chip-select logic can also generate  $\overline{DSACK}$  and  $\overline{AVEC}$  signals internally. Each signal can also be synchronized with the ECLK signal available on ADDR23.



CHIP SEL BLOCK

**Figure 11 Chip-Select Circuit Block Diagram**

When a memory access occurs, chip-select logic compares address space type, address, type of access, transfer size, and interrupt priority (in the case of interrupt acknowledge) to parameters stored in chip-select registers. If all parameters match, the appropriate chip-select signal is asserted. Select signals are active low. If a chip-select function is given the same address as a microcontroller module or an internal memory array, an access to that address goes to the module or array, and the chip-select signal is not asserted. The external address and data buses do not reflect the internal access.

All chip-select circuits except  $\overline{\text{CSBOOT}}$  are disabled out of reset. Chip-select option registers must not be written until base addresses have been written to the proper base address registers. Alternate functions for chip-select pins are enabled if appropriate data bus pins are held low at the release of  $\overline{\text{RESET}}$ .

**Table 17** lists allocation of chip-selects and discrete outputs on the pins of the MCU.

**Table 17 Chip-Select and Discrete Output Allocation**

Pin	Chip-Select	Discrete Outputs
$\overline{\text{CSBOOT}}$	$\overline{\text{CSBOOT}}$	—
BR	$\overline{\text{CS0}}$	—
$\overline{\text{BG}}$	$\overline{\text{CS1}}$	—
BGACK	$\overline{\text{CS2}}$	—
FC0	$\overline{\text{CS3}}$	PC0
FC1	$\overline{\text{CS4}}$	PC1
FC2	$\overline{\text{CS5}}$	PC2
ADDR19	$\overline{\text{CS6}}$	PC3
ADDR20	$\overline{\text{CS7}}$	PC4
ADDR21	$\overline{\text{CS8}}$	PC5
ADDR22	$\overline{\text{CS9}}$	PC6
ADDR23	$\overline{\text{CS10}}$	—

### 3.6.1 Chip-Select Registers

Each chip-select pin can have one or more functions. Chip-select pin assignment registers  $\text{CSPAR}[0:1]$  determine functions of the pins. Pin assignment registers also determine port size for dynamic bus allocation. A pin data register ( $\text{PORTC}$ ) latches data for chip-select pins that are used for discrete output.

Blocks of addresses are assigned to each chip-select function. Block sizes of two Kbytes to one Mbyte can be selected by writing values to the appropriate base address registers  $\text{CSBARBT}$  and  $\text{CSBAR}[0:10]$ . Multiple chip-selects assigned to the same block of addresses must have the same number of wait states.

Chip-select option registers  $\text{CSORBT}$  and  $\text{CSOR}[0:10]$  determine timing of and conditions for assertion of chip-select signals. Eight parameters, including operating mode, access size, synchronization, and wait state insertion can be specified.

Initialization software usually resides in a peripheral memory device controlled by the chip-select circuits.  $\overline{\text{CSBOOT}}$  and registers  $\text{CSORBT}$  and  $\text{CSBARBT}$  are provided to support bootstrap operation.

### 3.6.2 Pin Assignment Registers

The pin assignment registers contain twelve 2-bit fields that determine functions of the chip-select pins. Each pin has two or three possible functions, as shown in **Table 18**.

**Table 18 Chip-Select Pin Functions**

Assignment Register	16-Bit Chip-Select	Alternate Function	Discrete Output
CSPAR0	$\overline{\text{CSBOOT}}$	$\overline{\text{CSBOOT}}$	—
	$\overline{\text{CS0}}$	$\overline{\text{BR}}$	—
	$\overline{\text{CS1}}$	$\overline{\text{BG}}$	—
	$\overline{\text{CS2}}$	$\overline{\text{BGACK}}$	—
	$\overline{\text{CS3}}$	FC0	PC0
	$\overline{\text{CS4}}$	FC1	PC1
	$\overline{\text{CS5}}$	FC2	PC2
CSPAR1	$\overline{\text{CS6}}$	ADDR19	PC3
	$\overline{\text{CS7}}$	ADDR20	PC4
	$\overline{\text{CS8}}$	ADDR21	PC5
	$\overline{\text{CS9}}$	ADDR22	PC6
	$\overline{\text{CS10}}$	ADDR23	ECLK

**Table 19** shows pin assignment field encoding. Pins that have no discrete output function do not use the %00 encoding.

**Table 19 Pin Assignment Encodings**

Bit Field	Description
00	Discrete output
01	Alternate function
10	Chip-select (8-bit port)
11	Chip-select (16-bit port)

**CSPAR0** — Chip-Select Pin Assignment Register 0

**\$YFFA44**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	CS5PA[1:0]	CS4PA[1:0]	CS3PA[1:0]	CS2PA[1:0]	CS1PA[1:0]	CS0PA[1:0]	CSBTPA[1:0]							

RESET:

0	0	DATA2	1	DATA2	1	DATA2	1	DATA1	1	DATA1	1	DATA1	1	1	DATA0
---	---	-------	---	-------	---	-------	---	-------	---	-------	---	-------	---	---	-------

CSPAR0 contains seven 2-bit fields that determine the functions of corresponding chip-select pins. CSPAR0[15:14] are not used. These bits always read zero; writes have no effect. CSPAR0 bit 1 always reads one; writes to CSPAR0 bit 1 have no effect. **Table 20** shows CSPAR0 pin assignments.



**Table 20 CSPAR0 Pin Assignments**

CSPAR0 Field	Chip-Select Signal	Alternate Signal	Discrete Output
CS5PA[1:0]	CS5	FC2	PC2
CS4PA[1:0]	CS4	FC1	PC1
CS3PA[1:0]	CS3	FC0	PC0
CS2PA[1:0]	CS2	BGACK	—
CS1PA[1:0]	CS1	$\overline{BG}$	—
CS0PA[1:0]	CS0	$\overline{BR}$	—
CSBTPA[1:0]	$\overline{CSBOOT}$	—	—

**CSPAR1 — Chip-Select Pin Assignment Register 1**

**\$YFFA46**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	CS10PA[1:0]	CS9PA[1:0]	CS8PA[1:0]	CS7PA[1:0]	CS6PA[1:0]					

RESET:

0	0	0	0	0	0	DATA <sup>1</sup> <sub>7</sub>	1	DATA <sup>1</sup> <sub>[7:6]</sub>	1	DATA <sup>1</sup> <sub>[7:5]</sub>	1	DATA <sup>1</sup> <sub>[7:4]</sub>	1	DATA <sup>1</sup> <sub>[7:3]</sub>	1
---	---	---	---	---	---	--------------------------------	---	------------------------------------	---	------------------------------------	---	------------------------------------	---	------------------------------------	---

NOTES:

1. Refer to **Table 21** for CSPAR1 reset state information.

The reset state of DATA[7:3] determines whether pins controlled by CSPAR1 are initially configured as high-order address lines or chip-selects. **Table 21** shows the correspondence between DATA[7:3] and the reset configuration of  $\overline{CS}[10:6]$ /ADDR[23:19].

**Table 21 Reset Pin Function of  $\overline{CS}[10:6]$**

Data Bus Pins at Reset					Chip-Select/Address Bus Pin Function				
DATA7	DATA6	DATA5	DATA4	DATA3	$\overline{CS}10$ / ADDR23	$\overline{CS}9$ / ADDR22	$\overline{CS}8$ / ADDR21	$\overline{CS}7$ / ADDR20	$\overline{CS}6$ / ADDR19
1	1	1	1	1	$\overline{CS}10$	$\overline{CS}9$	$\overline{CS}8$	$\overline{CS}7$	$\overline{CS}6$
1	1	1	1	0	$\overline{CS}10$	$\overline{CS}9$	$\overline{CS}8$	$\overline{CS}7$	ADDR19
1	1	1	0	X	$\overline{CS}10$	$\overline{CS}9$	$\overline{CS}8$	ADDR20	ADDR19
1	1	0	X	X	$\overline{CS}10$	$\overline{CS}9$	ADDR21	ADDR20	ADDR19
1	0	X	X	X	$\overline{CS}10$	ADDR22	ADDR21	ADDR20	ADDR19
0	X	X	X	X	ADDR23	ADDR22	ADDR21	ADDR20	ADDR19

CSPAR1 contains five 2-bit fields that determine the functions of corresponding chip-select pins. CSPAR1[15:10] are not used. These bits always read zero; writes have no effect. **Table 22** shows CSPAR1 pin assignments.

**Table 22 CSPAR1 Pin Assignments**

CSPAR1 Field	Chip-Select Signal	Alternate Signal	Discrete Output
CS10PA[1:0]	$\overline{CS10}$	ADDR23	ECLK
CS9PA[1:0]	$\overline{CS9}$	ADDR22	PC6
CS8PA[1:0]	$\overline{CS8}$	ADDR21	PC5
CS7PA[1:0]	$\overline{CS7}$	ADDR20	PC4
CS6PA[1:0]	$\overline{CS6}$	ADDR19	PC3

Port size determines the way in which bus transfers to external addresses are allocated. Port size of eight bits or sixteen bits can be selected when a pin is assigned as a chip-select. Port size and transfer size affect how the chip-select signal is asserted. Refer to **3.6.4 Option Registers** for more information.

Out of reset, chip-select pin function is determined by the logic level on a corresponding data bus pin. These pins have weak internal pull-up drivers, but can be held low by external devices. Either 16-bit chip-select function (%11) or alternate function (%01) can be selected during reset. All pins except the boot ROM select pin ( $\overline{CSBOOT}$ ) are disabled out of reset.

The  $\overline{CSBOOT}$  signal is normally enabled out of reset. The state of the DATA0 line during reset determines what port width  $\overline{CSBOOT}$  uses. If DATA0 is held high (either by the weak internal pull-up driver or by an external pull-up device), 16-bit width is selected. If DATA0 is held low, 8-bit port size is selected.

A pin programmed as a discrete output drives an external signal to the value specified in the pin data register. No discrete output function is available on pins  $\overline{CSBOOT}$ , BR, BG, or  $\overline{BGACK}$ . ADDR23 provides ECLK output rather than a discrete output signal.

When a pin is programmed for discrete output or alternate function, internal chip-select logic still functions and can be used to generate  $\overline{DSACK}$  or  $\overline{AVEC}$  internally on an address and control signal match.

### 3.6.3 Base Address Registers

Each chip-select has an associated base address register. A base address is the lowest address in the block of addresses enabled by a chip-select. Block size is the extent of the address block above the base address. Block size is determined by the value contained in a BLKSZ field. Multiple chip-selects may be assigned to the same block of addresses so long as each chip-select uses the same number of wait states.

The BLKSZ field determines which bits in the base address field are compared to corresponding bits on the address bus during an access. Provided other constraints determined by option register fields are also satisfied, when a match occurs, the associated chip-select signal is asserted.

After reset, the MCU fetches the address of the first instruction to be executed from the reset vector, located beginning at address \$000000 in program space. To support bootstrap operation from reset, the base address field in CSBARBT has a reset value of all zeros. A memory device containing the reset vector and an initialization routine can be automatically enabled by  $\overline{CSBOOT}$  after a reset. The block size field in CSBARBT has a reset value of one Mbyte.

#### CSBARBT — Chip-Select Base Address Register Boot ROM \$YFFA48

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	BLKSZ[2:0]			
RESET:																
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

### CSBAR[0:10] — Chip-Select Base Address Registers

\$YFFA4C–\$YFFA74

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	BLKSZ[2:0]		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

#### ADDR[23:11] — Base Address Field

This field sets the starting address of a particular chip-select's address space. The address compare logic uses only the most significant bits to match an address within a block. The value of the base address must be a multiple of the block size. The base address register diagrams above show how register bits correspond to CPU address lines.

#### BLKSZ[2:0] — Block Size Field

This field determines the size of the block that must be enabled by the chip-select. **Table 23** shows bit encoding for the base address registers block size field.

**Table 23 Block Size Field Bit Encoding**

Block Size Field	Block Size	Address Lines Compared
000	2 Kbyte	ADDR[23:11]
001	8 Kbyte	ADDR[23:13]
010	16 Kbyte	ADDR[23:14]
011	64 Kbyte	ADDR[23:16]
100	128 Kbyte	ADDR[23:17]
101	256 Kbyte	ADDR[23:18]
110	512 Kbyte	ADDR[23:19]
111	1 Mbyte	ADDR[23:20]

### 3.6.4 Option Registers

The option registers contain eight fields that determine timing of and conditions for assertion of chip-select signals. To assert a chip-select signal, and to provide  $\overline{DSACK}$  or autovector support, other constraints set by fields in the option register and in the base address register must also be satisfied.

#### CSORBT — Chip-Select Option Register Boot ROM

\$YFFA4A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE	BYTE[1:0]		R $\overline{W}$ [1:0]		STRB	$\overline{DSACK}$ [3:0]			SPACE[1:0]		IPL[2:0]		$\overline{AVEC}$		
RESET:															
0	1	1	1	1	0	1	1	0	1	1	1	0	0	0	0

#### CSOR[0:10] — Chip-Select Option Registers

\$YFFA4E–YFFA76

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE	BYTE[1:0]		R $\overline{W}$ [1:0]		STRB	$\overline{DSACK}$ [3:0]			SPACE[1:0]		IPL[2:0]		$\overline{AVEC}$		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSORBT, the option register for  $\overline{\text{CSBOOT}}$ , contains special reset values that support bootstrap operation from peripheral memory devices.

The following bit descriptions apply to both CSORBT and CSOR[0:10] option registers.

**MODE** — Asynchronous/Synchronous Mode

0 = Asynchronous mode (chip-select assertion determined by bus control signals)

1 = Synchronous mode (chip-select assertion synchronized with ECLK signal)

In asynchronous mode, the chip-select is asserted synchronized with  $\overline{\text{AS}}$  or  $\overline{\text{DS}}$ .

$\overline{\text{DSACK}}[3:0]$  is not used in synchronous mode because a bus cycle is only performed as a synchronous operation. When a match condition occurs on a chip-select programmed for synchronous operation, the chip-select signals the EBI that an ECLK cycle is pending.

**BYTE[1:0]** — Upper/Lower Byte Option

This field is used only when the chip-select 16-bit port option is selected in the pin assignment register.

**Table 24** lists upper/lower byte options.

**Table 24 Upper/Lower Byte Options**

BYTE[1:0]	Description
00	Disable
01	Lower Byte
10	Upper Byte
11	Both Bytes

**R/W[1:0]** — Read/Write

This field causes a chip-select to be asserted only for reads, only for writes, or for both reads and writes.

Refer to **Table 25** for options available.

**Table 25 R/W Encodings**

R/W[1:0]	Description
00	Reserved
01	Read Only
10	Write Only
11	Read/Write

**STRB** — Address Strobe/Data Strobe

0 = Address strobe

1 = Data strobe

This bit controls the timing for assertion of a chip-select in asynchronous mode. Selecting address strobe causes chip-select to be asserted synchronized with address strobe. Selecting data strobe causes chip-select to be asserted synchronized with data strobe.

**DSACK[3:0]** — Data and Size Acknowledge

This field specifies the source of  $\overline{\text{DSACK}}[3:0]$  in asynchronous mode. It also allows the user to adjust bus timing with internal  $\overline{\text{DSACK}}[3:0]$  generation by controlling the number of wait states that are inserted to optimize bus speed in a particular application. **Table 26** shows the  $\overline{\text{DSACK}}[3:0]$  encoding. The fast termination encoding (%1110) is used for two-cycle access to external memory.

**Table 26 DSACK Field Encoding**

<b>DSACK[3:0]</b>	<b>Clock Cycles Required Per Access</b>	<b>Wait States Per Access</b>
0000	3	0 Wait States
0001	4	1 Wait State
0010	5	2 Wait States
0011	6	3 Wait States
0100	7	4 Wait States
0101	8	5 Wait States
0110	9	6 Wait States
0111	10	7 Wait States
1000	11	8 Wait States
1001	12	9 Wait States
1010	13	10 Wait States
1011	14	11 Wait States
1100	15	12 Wait States
1101	16	13 Wait States
1110	2	Fast Termination
1111	—	External DSACK

**SPACE[1:0] — Address Space**

Use this option field to select an address space for the chip-select logic. The CPU32L normally operates in supervisor or user space, but interrupt acknowledge cycles must take place in CPU space. **Table 27** shows address space bit encodings.

**Table 27 Address Space Bit Encodings**

<b>SPACE[1:0]</b>	<b>Address Space</b>
00	CPU Space
01	User Space
10	Supervisor Space
11	Supervisor/User Space

**IPL[2:0] — Interrupt Priority Level**

If the space field is set for CPU space, chip-select logic can be used for interrupt acknowledge. During an interrupt acknowledge cycle, the priority level on address lines ADDR[3:1] is compared to the value in IPL[2:0]. If the values are the same, a chip-select is asserted, provided that other option register conditions are met. **Table 28** shows IPL[2:0] encoding.

**Table 28 Interrupt Priority Level Field Encoding**

IPL[2:0]	Interrupt Priority Level
000	Any Level
001	1
010	2
011	3
100	4
101	5
110	6
111	7

This field only affects the response of chip-selects and does not affect interrupt recognition by the CPU. Any level means that chip-select is asserted regardless of the level of the interrupt acknowledge cycle.

**$\overline{AVEC}$  — Autovector Enable**

- 0 = External interrupt vector enabled
- 1 = Autovector enabled

This field selects one of two methods of acquiring an interrupt vector number during an external interrupt acknowledge cycle.

If the chip-select is configured to trigger on an interrupt acknowledge cycle ( $SPACE[1:0] = \%00$ ) and the  $\overline{AVEC}$  field is set to one, the chip-select circuit generates an internal  $\overline{AVEC}$  signal in response to an external interrupt cycle, and the SIML supplies an automatic vector number. Otherwise, the vector number must be supplied by the requesting device. An internal autovector is generated only in response to interrupt requests from the SIML  $\overline{IRQ}$  pins. Interrupt requests from other IMB modules are ignored.

The  $\overline{AVEC}$  bit must not be used in synchronous mode, as autovector response timing can vary because of ECLK synchronization.

**3.6.5 Port C Data Register**

Bit values in port C determine the state of chip-select pins used for discrete output. When a pin is assigned as a discrete output, the value in this register appears at the output. This is a read/write register. Bit 7 is not used. Writing to this bit has no effect, and it always returns zero when read.

**PORTC — Port C Data Register**

**\$YFFA41**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								0	PC6	PC5	PC4	PC3	PC2	PC1	PC0

RESET:

0 1 1 1 1 1 1 1

**3.7 General-Purpose Input/Output**

SIML pins can be configured as two general-purpose I/O ports, E and F. The following paragraphs describe registers that control the ports.

**PORTE0, PORTE1— Port E Data Register****\$YFFA11, YFFA13**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0

RESET:

U    U    U    U    U    U    U    U

A write to the port E data register is stored in the internal data latch and, if any port E pin is configured as an output, the value stored for that bit is driven on the pin. A read of the port E data register returns the value at the pin only if the pin is configured as a discrete input. Otherwise, the value read is the value stored in the register.

The port E data register is a single register that can be accessed in two locations. When accessed at \$YFFA11, the register is referred to as PORTE0; when accessed at \$YFFA13, the register is referred to as PORTE1. The register can be read or written at any time. It is unaffected by reset.

**DDRE — Port E Data Direction Register****\$YFFA15**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0

RESET:

0    0    0    0    0    0    0    0

The bits in this register control the direction of the pin drivers when the pins are configured as I/O. 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. This register can be read or written at any time.

**PEPAR — Port E Pin Assignment****\$YFFA17**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								PEPA7	PEPA6	PEPA5	PEPA4	PEPA3	PEPA2	PEPA1	PEPA0

RESET:

DATA8 DATA8 DATA8 DATA8 DATA8 DATA8 DATA8 DATA8

The bits in this register control the function of each port E pin. Any bit set to one configures the corresponding pin as a bus control signal, with the function shown in **Table 29**. Any bit cleared to zero defines the corresponding pin to be an I/O pin, controlled by PORTE and DDRE.

Data bus bit 8 controls the state of this register following reset. If DATA8 is set to one during reset, the register is set to \$FF, which defines all port E pins as bus control signals. If DATA8 is cleared to zero during reset, this register is set to \$00, configuring all port E pins as I/O pins.

**Table 29 Port E Pin Assignments**

PEPAR Bit	Port E Signal	Bus Control Signal
PEPA7	PE7	SIZ1
PEPA6	PE6	SIZ0
PEPA5	PE5	$\overline{AS}$
PEPA4	PE4	$\overline{DS}$
PEPA3	PE3	$\overline{RMC}$
PEPA2	PE2	$\overline{AVEC}$
PEPA1	PE1	$\overline{DSACK1}$
PEPA0	PE0	$\overline{DSACK0}$

**PORTF0, PORTF1 — Port F Data Register**

**\$YFFA19, YFFA1B**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
RESET:								U	U	U	U	U	U	U	U

The write to the port F data register is stored in the internal data latch, and if any port F pin is configured as an output, the value stored for that bit is driven onto the pin. A read of the port F data register returns the value at the pin only if the pin is configured as a discrete input. Otherwise, the value read is the value stored in the register.

The port F data register is a single register that can be accessed in two locations. When accessed at \$YFFA19, the register is referred to as PORTF0; when accessed at \$YFFA1B, the register is referred to as PORTF1. The register can be read or written at any time. It is unaffected by reset.

**DDRF — Port F Data Direction Register**

**\$YFFA1D**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0
RESET:								0	0	0	0	0	0	0	0

The bits in this register control the direction of the pin drivers when the pins are configured as I/O. 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. This register can be read or written at any time.

**PFPAR — Port F Pin Assignment Register**

**\$YFFA1F**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								PFFA7	PFFA6	PFFA5	PFFA4	PFFA3	PFFA2	PFFA1	PFFA0
RESET:								DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9

The bits in this register control the function of each port F pin. Any bit cleared to zero defines the corresponding pin to be an I/O pin. Any bit set to one defines the corresponding pin to be an interrupt request signal or MODCLK. The MODCLK signal has no function after reset. **Table 30** shows port F pin assignments.



**Table 30 Port F Pin Assignments**

PFPA Field	Port F Signal	Alternate Signal
PFPA7	PF7	$\overline{\text{IRQ7}}$
PFPA6	PF6	$\overline{\text{IRQ6}}$
PFPA5	PF5	$\overline{\text{IRQ5}}$
PFPA4	PF4	$\overline{\text{IRQ4}}$
PFPA3	PF3	$\overline{\text{IRQ3}}$
PFPA2	PF2	$\overline{\text{IRQ2}}$
PFPA1	PF1	$\overline{\text{IRQ1}}$
PFPA0	PF0	MODCLK

Data bus pin 9 controls the state of this register following reset. If DATA9 is set to one during reset, the register is set to \$FF, which defines all port F pins as interrupt request inputs. If DATA9 is cleared to zero during reset, this register is set to \$00, defining all port F pins as I/O pins.

### 3.8 Resets

Reset procedures handle system initialization and recovery from catastrophic failure. The MCU performs resets with a combination of hardware and software. The SIML determines whether a reset is valid, asserts control signals, performs basic system configuration based on hardware mode-select inputs, then passes control to the CPU.

Reset occurs when an active low logic level on the  $\overline{\text{RESET}}$  pin is clocked into the SIML. Resets are gated by the CLKOUT signal. Asynchronous resets are assumed to be catastrophic. An asynchronous reset can occur on any clock edge. Synchronous resets are timed to occur at the end of bus cycles. If there is no clock when  $\overline{\text{RESET}}$  is asserted, reset does not occur until the clock starts. Resets are clocked in order to allow completion of write cycles in progress at the time  $\overline{\text{RESET}}$  is asserted.

Reset is the highest-priority CPU32L exception. Any processing in progress is aborted by the reset exception, and cannot be restarted. Only essential tasks are performed during reset exception processing. Other initialization tasks must be accomplished by the exception handler routine.

#### 3.8.1 SIML Reset Mode Selection

The logic states of certain data bus pins during reset determine SIML operating configuration. In addition, the state of the MODCLK pin determines system clock source and the state of the BKPT pin determines what happens during subsequent breakpoint assertions. **Table 31** is a summary of reset mode selection options.

**Table 31 Reset Mode Selection**

Mode Select Pin	Default Function (Pin Left High)	Alternate Function (Pin Pulled Low)
DATA0	$\overline{\text{CSBOOT}}$ 16-Bit	$\overline{\text{CSBOOT}}$ 8-Bit
DATA1	$\overline{\text{CS0}}$ $\overline{\text{CS1}}$ $\overline{\text{CS2}}$	$\overline{\text{BR}}$ $\overline{\text{BG}}$ $\overline{\text{BGACK}}$
DATA2	$\overline{\text{CS3}}$ $\overline{\text{CS4}}$ $\overline{\text{CS5}}$	FC0 FC1 FC2
DATA3 DATA4 DATA5 DATA6 DATA7	$\overline{\text{CS6}}$ $\overline{\text{CS}}[7:6]$ $\overline{\text{CS}}[8:6]$ $\overline{\text{CS}}[9:6]$ $\overline{\text{CS}}[10:6]$	ADDR19 ADDR[20:19] ADDR[21:19] ADDR[22:19] ADDR[23:19]
DATA8	$\overline{\text{DSACK}}[1:0]$ AVEC, DS, AS $\overline{\text{SIZ}}[1:0]$	PORTE
DATA9	$\overline{\text{IRQ}}[7:1]$ MODCLK	PORTF
DATA11	Test Mode Disabled	Test Mode Enabled
MODCLK	VCO = System Clock	EXTAL = System Clock
$\overline{\text{BKPT}}$	Background Mode Disabled	Background Mode Enabled

Data lines have weak internal pull-up drivers. External bus loading can overcome the weak internal pull-up drivers on data bus lines, and hold pins low during reset. Use an active device to hold data bus lines low. Data bus configuration logic must release the bus before the first bus cycle after reset to prevent conflict with external memory devices. The first bus cycle occurs ten CLKOUT cycles after  $\overline{\text{RESET}}$  is released. If external mode selection logic causes a conflict of this type, an isolation resistor on the driven lines may be required.

### 3.8.2 Reset States of SIML Pins

Generally, while  $\overline{\text{RESET}}$  is asserted, SIML pins either go to an inactive high-impedance state or are driven to their inactive states. After  $\overline{\text{RESET}}$  is released, mode selection occurs and reset exception processing begins. Pins configured as inputs must be driven to the desired active state. Pull-up or pull-down circuitry may be necessary. Pins configured as outputs begin to function after  $\overline{\text{RESET}}$  is released. **Table 32** is a summary of SIML pin states during reset.

**Table 32 SIML Pin Reset States**

Pin(s)	Pin State While RESET Asserted	Pin State After RESET Released			
		Default Function		Alternate Function	
		Pin Function	Pin State	Pin Function	Pin State
$\overline{CS10}/ADDR23/ECLK$	$V_{DD}$	$\overline{CS10}$	$V_{DD}$	ADDR23	Unknown
$\overline{CS}[9:6]/ADDR[22:19]/PC[6:3]$	$V_{DD}$	$\overline{CS}[9:6]$	$V_{DD}$	ADDR[22:19]	Unknown
ADDR[18:0]	High-Z	ADDR[18:0]	Unknown	ADDR[18:0]	Unknown
$\overline{AS}/PE5$	High-Z	$\overline{AS}$	Output	PE5	Input
$\overline{AVEC}/PE2$	High-Z	$\overline{AVEC}$	Input	PE2	Input
$\overline{BERR}$	High-Z	$\overline{BERR}$	Input	$\overline{BERR}$	Input
$\overline{CS1}/BG$	$V_{DD}$	$\overline{CS1}$	$V_{DD}$	$\overline{BG}$	$V_{DD}$
$\overline{CS2}/BGACK$	$V_{DD}$	$\overline{CS2}$	$V_{DD}$	$\overline{BGACK}$	Input
$\overline{CS0}/BR$	$V_{DD}$	$\overline{CS0}$	$V_{DD}$	$\overline{BR}$	Input
CLKOUT	Output	CLKOUT	Output	CLKOUT	Output
$\overline{CSBOOT}$	$V_{DD}$	$\overline{CSBOOT}$	$V_{SS}$	$\overline{CSBOOT}$	$V_{SS}$
DATA[15:0]	Mode select	DATA[15:0]	Input	DATA[15:0]	Input
$\overline{DS}/PE4$	High-Z	$\overline{DS}$	Output	PE4	Input
$\overline{DSACK0}/PE0$	High-Z	$\overline{DSACK0}$	Input	PE0	Input
$\overline{DSACK1}/PE1$	High-Z	$\overline{DSACK1}$	Input	PE1	Input
$\overline{CS}[5:3]/FC[2:0]/PC[2:0]$	$V_{DD}$	$\overline{CS}[5:3]$	$V_{DD}$	FC[2:0]	Unknown
$\overline{HALT}$	High-Z	$\overline{HALT}$	Input	$\overline{HALT}$	Input
$\overline{IRQ}[7:1]/PF[7:1]$	High-Z	$\overline{IRQ}[7:1]$	Input	PF[7:1]	Input
MODCLK/PF0	Mode Select	MODCLK	Input	PF0	Input
$R/\overline{W}$	High-Z	$R/\overline{W}$	Output	$R/\overline{W}$	Output
$\overline{RESET}$	Asserted	$\overline{RESET}$	Input	$\overline{RESET}$	Input
$\overline{RMC}/PE3$	High-Z	$\overline{RMC}$	Output	PE3	Input
SIZ[1:0]/PE[7:6]	High-Z	SIZ[1:0]	Unknown	PE[7:6]	Input
TSC	Mode select	TSC	Input	TSC	Input

### 3.8.3 Functions of Pins for Other Modules During Reset

Generally, pins associated with modules other than the SIML default to port functions, and input/output ports are set to input state. This is accomplished by disabling pin functions in the appropriate control registers, and by clearing the appropriate port data direction registers. Refer to individual module sections in this manual for more information.

### 3.8.4 Reset Timing

The  $\overline{RESET}$  input must be asserted for a specified minimum period in order for reset to occur. External  $\overline{RESET}$  assertion can be delayed internally for a period equal to the longest bus cycle time (or the bus monitor time-out period) in order to protect write cycles from being aborted by reset. While  $\overline{RESET}$  is asserted, SIML pins are either in a disabled high-impedance state or are driven to their inactive states.

When an external device asserts  $\overline{RESET}$  for the proper period, reset control logic clocks the signal into an internal latch. The control logic drives the  $\overline{RESET}$  pin low for an additional 512 CLKOUT cycles after it detects that the  $\overline{RESET}$  signal is no longer being externally driven, to guarantee this length of reset to the entire system.

If an internal source asserts the reset signal, the reset control logic asserts  $\overline{\text{RESET}}$  for a minimum of 512 cycles. If the reset signal is still asserted at the end of 512 cycles, the control logic continues to assert  $\overline{\text{RESET}}$  until the internal reset signal is negated.

After 512 cycles have elapsed, the reset input pin goes to an inactive, high-impedance state for 10 cycles. At the end of this 10-cycle period, the reset input is tested. When the input is at logic level one, reset exception processing begins. If, however, the reset input is at logic level zero, the reset control logic drives the pin low for another 512 cycles. At the end of this period, the pin again goes to high-impedance state for ten cycles, then it is tested again. The process repeats until  $\overline{\text{RESET}}$  is released.

### 3.8.5 Power-On Reset

When the SIML clock synthesizer is used to generate the system clock, power-on reset involves special circumstances related to application of system and clock synthesizer power. Regardless of clock source, voltage must be applied to clock synthesizer power input pin  $V_{\text{DDSYN}}$  in order for the MCU to operate. The following discussion assumes that  $V_{\text{DDSYN}}$  is applied before and during reset. This minimizes crystal start-up time. When  $V_{\text{DDSYN}}$  is applied at power-on, start-up time is affected by specific crystal parameters and by oscillator circuit design.  $V_{\text{DD}}$  ramp-up time also affects pin state during reset.

During power-on reset, an internal circuit in the SIML drives the IMB and external reset lines. The circuit releases the internal reset line as  $V_{\text{DD}}$  ramps up to the minimum specified value, and SIML pins are initialized. As  $V_{\text{DD}}$  reaches a specified minimum value, the clock synthesizer VCO begins operation and clock frequency ramps up to specified limp mode frequency. The external  $\overline{\text{RESET}}$  line remains asserted until the clock synthesizer PLL locks and 512 CLKOUT cycles elapse.

The SIML clock synthesizer provides clock signals to the other MCU modules. After the clock is running and the internal reset signal is asserted for four clock cycles, these modules reset.  $V_{\text{DD}}$  ramp time and VCO frequency ramp time determine how long these four cycles take. Worst case is approximately 15 milliseconds. During this period, module port pins may be in an indeterminate state. While input-only pins can be put in a known state by means of external pull-up resistors, external logic on input/output or output-only pins must condition the lines during this time. Active drivers require high-impedance buffers or isolation resistors to prevent conflict.

### 3.8.6 Use of Three-State Control Pin

Asserting the three-state control (TSC) input causes the MCU to put all output drivers in an inactive, high-impedance state. The signal must remain asserted for ten clock cycles in order for drivers to change state. There are certain constraints on use of TSC during power-on reset:

- When the internal clock synthesizer is used (MODCLK held high during reset), synthesizer ramp-up time affects how long the ten cycles take. Worst case is approximately 20 ms from TSC assertion.
- When an external clock signal is applied (MODCLK held low during reset), pins go to high-impedance state as soon after TSC assertion as ten clock pulses have been applied to the EXTAL pin.
- When TSC assertion takes effect, internal signals are forced to values that can cause inadvertent mode selection. Once the output drivers change state, the MCU must be powered down and restarted before normal operation can resume.

## 3.9 Interrupts

Interrupt recognition and servicing involve complex interaction between the CPU32L, the SIML, and a device or module requesting interrupt service.

The CPU32L provides seven levels of interrupt priority (1–7), seven automatic interrupt vectors, and 200 assignable interrupt vectors. All interrupts with priorities less than seven can be masked by the interrupt priority (IP) field in status register. The CPU32L handles interrupts as a type of asynchronous expression.

There are seven interrupt request signals ( $\overline{\text{IRQ}}[7:1]$ ). These signals are used internally on the IMB, and there are corresponding pins for external interrupt service requests. The CPU treats all interrupt requests as though they come from internal modules — external interrupt requests are treated as interrupt service requests from the SIML. Each of the interrupt request signals corresponds to an interrupt priority level.  $\overline{\text{IRQ}}1$  has the lowest priority and  $\overline{\text{IRQ}}7$  the highest.

Interrupt recognition is determined by interrupt priority level and interrupt priority mask value. The interrupt priority mask consists of three bits in the CPU32L status register. Binary values %000 to %111 provide eight priority masks. Masks prevent an interrupt request of a priority less than or equal to the mask value from being recognized and processed.  $\overline{\text{IRQ}}7$ , however, is always recognized, even if the mask value is %111.

$\overline{\text{IRQ}}[7:1]$  are active-low level-sensitive inputs. The low on the pin must remain asserted until an interrupt acknowledge cycle corresponding to that level is detected.

$\overline{\text{IRQ}}7$  is transition-sensitive as well as level-sensitive: a level 7 interrupt is not detected unless a falling edge transition is detected on the  $\overline{\text{IRQ}}7$  line. This prevents redundant servicing and stack overflow. A non-maskable interrupt is generated each time  $\overline{\text{IRQ}}7$  is asserted as well as each time the priority mask changes from %111 to a lower number while  $\overline{\text{IRQ}}7$  is asserted.

Interrupt requests are sampled on consecutive falling edges of the system clock. Interrupt request input circuitry has hysteresis: to be valid, a request signal must be asserted for at least two consecutive clock periods. Valid requests do not cause immediate exception processing, but are left pending. Pending requests are processed at instruction boundaries or when exception processing of higher-priority exceptions is complete.

The CPU32L does not latch the priority of a pending interrupt request. If an interrupt source of higher priority makes a service request while a lower priority request is pending, the higher priority request is serviced. If an interrupt request with a priority equal to or lower than the current IP mask value is made, the CPU32L does not recognize the occurrence of the request. If simultaneous interrupt requests of different priorities are made, and both have a priority greater than the mask value, the CPU32L recognizes the higher-level request.

### 3.9.1 Interrupt Acknowledge and Arbitration

When the CPU32L detects one or more interrupt requests of a priority higher than the interrupt priority mask value, it places the interrupt request level on the address bus and initiates a CPU space read cycle. The request level serves two purposes: it is decoded by modules or external devices that have requested interrupt service, to determine whether the current interrupt acknowledge cycle pertains to them, and it is latched into the interrupt priority mask field in the CPU32L status register, to preclude further interrupts of lower priority during interrupt service.

Modules or external devices that have requested interrupt service must decode the interrupt priority mask value placed on the address bus during the interrupt acknowledge cycle and respond if the priority of the service request corresponds to the mask value. However, before modules or external devices respond, interrupt arbitration takes place.

Arbitration is performed by means of serial contention between values stored in individual module interrupt arbitration (IARB) fields. Each module that can make an interrupt service request, including the SIML, has an IARB field in its configuration register. IARB fields can be assigned values from %0000 to %1111. In order to implement an arbitration scheme, each module that can initiate an interrupt service request must be assigned a unique, non-zero IARB field value during system initialization. Arbitration priorities range from %0001 (lowest) to %1111 (highest) — if the CPU recognizes an interrupt service request from a source that has an IARB field value of %0000, a spurious interrupt exception is processed.

### WARNING

Do not assign the same arbitration priority to more than one module. When two or more IARB fields have the same non-zero value, the CPU32L interprets multiple vector numbers at the same time, with unpredictable consequences.

Because the EBI manages external interrupt requests, the SIML IARB value is used for arbitration between internal and external interrupt requests. The reset value of IARB for the SIML is %1111, and the reset IARB value for all other modules is %0000.

Although arbitration is intended to deal with simultaneous requests of the same priority, it always takes place, even when a single source is requesting service. This is important for two reasons: the EBI does not transfer the interrupt acknowledge read cycle to the external bus unless the SIML wins contention, and failure to contend causes the interrupt acknowledge bus cycle to be terminated early, by a bus error.

When arbitration is complete, the module with the highest arbitration priority must terminate the bus cycle. Internal modules place an interrupt vector number on the data bus and generate appropriate internal cycle termination signals. In the case of an external interrupt request, after the interrupt acknowledge cycle is transferred to the external bus, the appropriate external device must decode the mask value and respond with a vector number, then generate data and size acknowledge ( $\overline{DSACK}$ ) termination signals, or it must assert the autovector ( $\overline{AVEC}$ ) request signal. If the device does not respond in time, the EBI bus monitor asserts the bus error signal ( $\overline{BERR}$ ), and a spurious interrupt exception is taken.

Chip-select logic can also be used to generate internal  $\overline{AVEC}$  or  $\overline{DSACK}$  signals in response to interrupt requests from external devices. Chip-select address match logic functions only after the EBI transfers an interrupt acknowledge cycle to the external bus following IARB contention. If a module makes an interrupt request of a certain priority, and the appropriate chip-select registers are programmed to generate  $\overline{AVEC}$  or  $\overline{DSACK}$  signals in response to an interrupt acknowledge cycle for that priority level, chip-select logic does not respond to the interrupt acknowledge cycle, and the internal module supplies a vector number and generates internal cycle termination signals.

For periodic timer interrupts, the PIRQL field in the periodic interrupt control register (PICR) determines PIT priority level. A PIRQL value of %000 means that PIT interrupts are inactive. By hardware convention, when the CPU32L receives simultaneous interrupt requests of the same level from more than one SIML source (including external devices), the periodic interrupt timer is given the highest priority, followed by the  $\overline{IRQ}$  pins. Refer to **3.4.6 Periodic Interrupt Timer** for more information.

### 3.9.2 Interrupt Processing Summary

A summary of the interrupt processing sequence follows. When the sequence begins, a valid interrupt service request has been detected and is pending.

- A. The CPU finishes higher priority exception processing or reaches an instruction boundary.
- B. The processor state is stacked. The S bit in the status register is set, establishing supervisor access level, and bits T1 and T0 are cleared, disabling tracing.
- C. The interrupt acknowledge cycle begins:
  - 1. FC[2:0] are driven to %111 (CPU space) encoding.
  - 2. The address bus is driven as follows: ADDR[23:20] = %1111; ADDR[19:16] = %1111, which indicates that the cycle is an interrupt acknowledge CPU space cycle; ADDR[15:4] = %111111111111; ADDR[3:1] = the priority of the interrupt request being acknowledged; and ADDR0 = %1.
  - 3. The request level is latched from the address bus into the interrupt priority mask field in the status or condition code register.
- D. Modules that have requested interrupt service decode the priority value on ADDR[3:1]. If request priority is the same as acknowledged priority, arbitration by IARB contention takes place.
- E. After arbitration, the interrupt acknowledge cycle is completed in one of the following ways:
  - 1. When there is no contention (IARB = %0000), the spurious interrupt monitor asserts  $\overline{\text{BERR}}$ , and the CPU generates the spurious interrupt vector number.
  - 2. The dominant interrupt source supplies a vector number and  $\overline{\text{DSACK}}$  signals appropriate to the access. The CPU acquires the vector number.
  - 3. The  $\overline{\text{AVEC}}$  signal is asserted (the signal can be asserted by the dominant interrupt source or the pin can be tied low), and the CPU generates an autovector number corresponding to interrupt priority.
  - 4. The bus monitor asserts  $\overline{\text{BERR}}$  and the CPU32L generates the spurious interrupt vector number.
- F. The vector number is converted to a vector address.
- G. The content of the vector address is loaded into the PC, and the processor transfers control to the exception handler routine.

### 3.10 Factory Test Block

The test submodule supports scan-based testing of the various MCU modules. It is integrated into the SIML to support production testing.

Test submodule registers are intended for Motorola use. Register names and addresses are provided to indicate that these addresses are occupied.

<b>SIMLTR</b> — System Integration Module Test Register	<b>\$YFFA02</b>
<b>SIMLTRE</b> — System Integration Module Test Register (E Clock)	<b>\$YFFA08</b>
<b>TSTMSRA</b> — Master Shift Register A	<b>\$YFFA30</b>
<b>TSTMSRB</b> — Master Shift Register B	<b>\$YFFA32</b>
<b>TSTSC</b> — Test Module Shift Count	<b>\$YFFA34</b>
<b>TSTRC</b> — Test Module Repetition Count	<b>\$YFFA36</b>
<b>CREG</b> — Test Module Control Register	<b>\$YFFA38</b>
<b>DREG</b> — Test Module Distributed Register	<b>\$YFFA3A</b>

## 4 Low-Power Central Processor Unit

Based on the powerful MC68020, the CPU32L processing module provides enhanced system performance and also uses the extensive software base for the Motorola M68000 family.

### 4.1 Overview

The CPU32L is fully object-code compatible with the M68000 family, which excels at processing calculation-intensive algorithms and supporting high-level languages. The CPU32L supports all of the MC68010 and most of the MC68020 enhancements, such as virtual memory support, loop mode operation, instruction pipeline, and 32-bit mathematical operations. Powerful addressing modes provide compatibility with existing software programs and increase the efficiency of high-level language compilers. Special instructions, such as table lookup and interpolate and low-power stop, support the specific requirements of controller applications. Also included is background debug mode, an alternate operating mode that suspends normal operation and allows the CPU to accept debugging commands from the development system.

Ease of programming is an important consideration in using a microcontroller. The CPU32L instruction set is optimized for high performance. The eight 32-bit general-purpose data registers readily support 8-bit (byte), 16-bit (word), and 32-bit (long word) operations. Ease of program checking and diagnosis is further enhanced by trace and trap capabilities at the instruction level.

Use of high-level languages is increasing as controller applications become more complex and control programs become larger. High-level languages aid rapid development of software, with less error, and are readily portable. The CPU32L instruction set supports high-level languages.

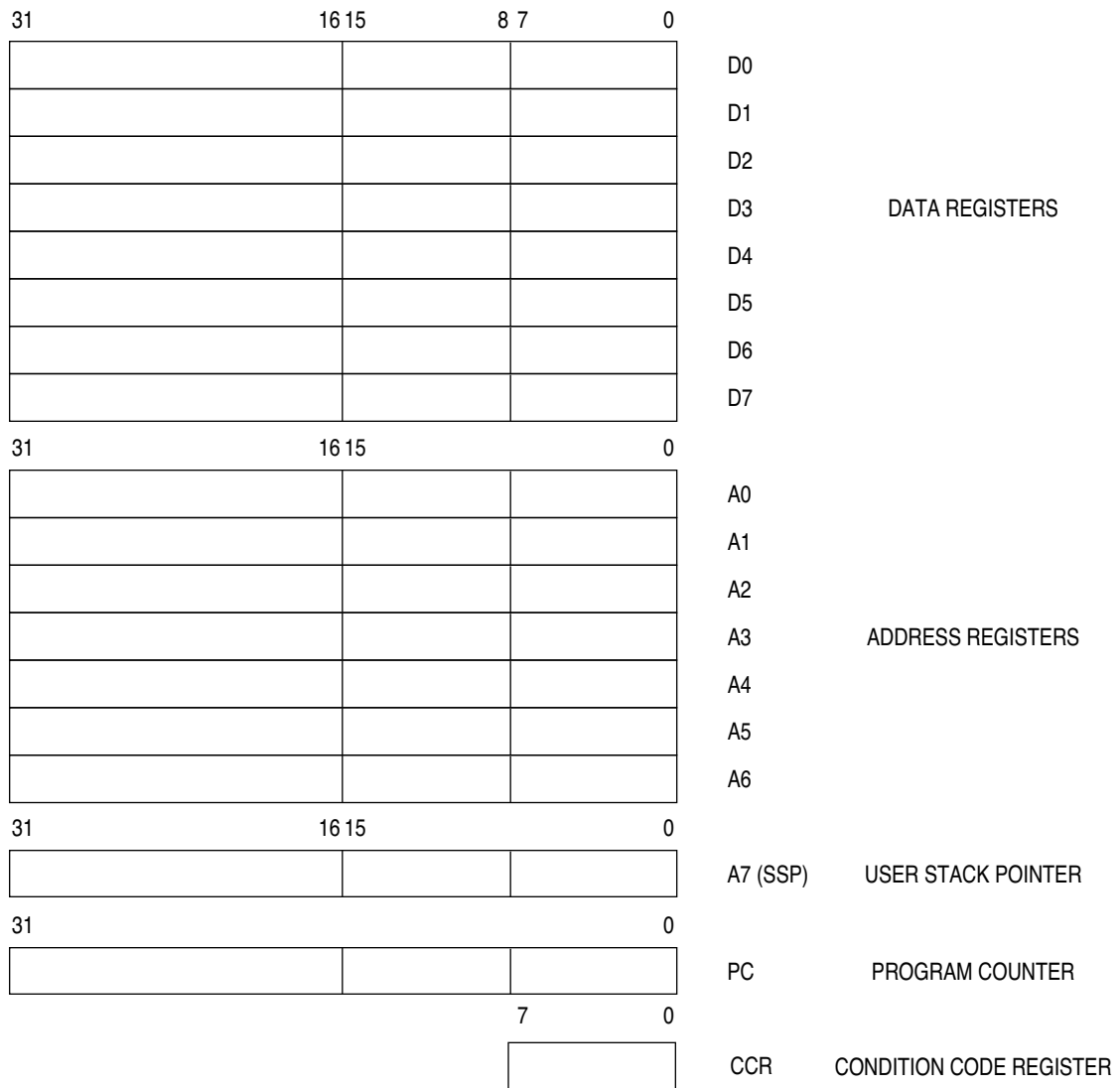
### 4.2 Programming Model

The CPU32L has sixteen 32-bit general registers, a 32-bit program counter, one 32-bit supervisor stack pointer, a 16-bit status register, two alternate function code registers, and a 32-bit vector base register.

The programming model of the CPU32L consists of a user model shown in **Figure 12** and a supervisor model shown in **Figure 13**, corresponding to the user and supervisor privilege levels. Some instructions available at the supervisor level are not available at the user level, allowing the supervisor to protect system resources from uncontrolled access. Bit S in the status register determines the privilege level.

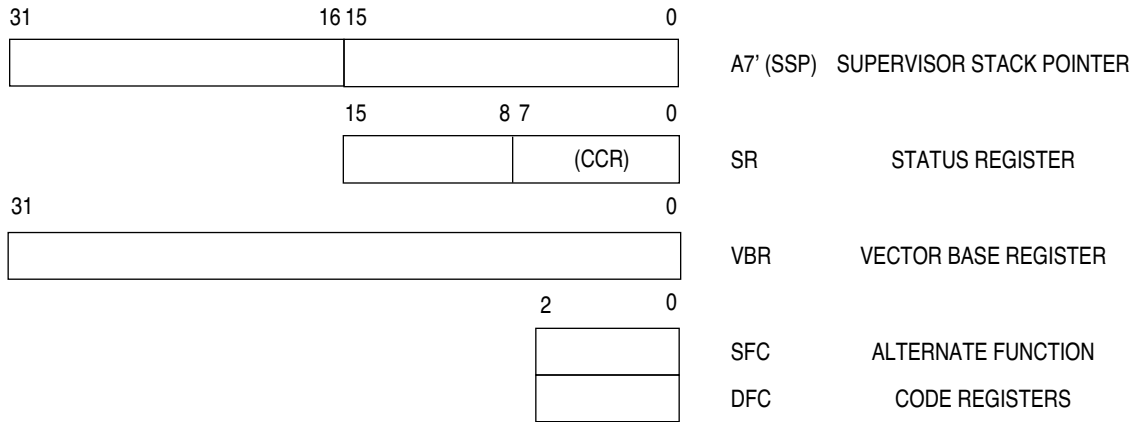
The user programming model remains unchanged from previous M68000 family microprocessors. Application software written to run at the nonprivileged user level migrates without modification to the CPU32L from any M68000 platform. The move from SR instruction, however, is privileged in the CPU32L. It is not privileged in the M68000.





CPU32 USER PROG MODEL

**Figure 12 User Programming Model**



CPU32 SUPV PROG MODEL

**Figure 13 Supervisor Programming Model Supplement**

### 4.3 Status Register

The status register contains the condition codes that reflect the results of a previous operation and can be used for conditional instruction execution in a program. The lower byte containing the condition codes is the only portion of the register available at the user privilege level; it is referenced as the condition code register (CCR) in user programs. At the supervisor privilege level, software can access the full status register, including the interrupt priority mask and additional control bits.

#### SR — Status Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T1	T0	S	0	0	IP			0	0	0	X	N	Z	V	C
RESET:															
0	0	1	0	0	1	1	1	0	0	0	U	U	U	U	U

#### System Byte

- T[1:0] — Trace Enable
- S — Supervisor/User State
- Bits [12:11] — Unimplemented
- IP[2:0] — Interrupt Priority Mask

#### User Byte (Condition Code Register)

- Bits [7:5] — Unimplemented
- X — Extend
- N — Negative
- Z — Zero
- V — Overflow
- C — Carry

#### 4.4 Data Types

Six basic data types are supported:

- Bits
- Packed Binary Coded Decimal Digits
- Byte Integers (8 bits)
- Word Integers (16 bits)
- Long Word Integers (32 bits)
- Quad Word Integers (64 bits)

#### 4.5 Addressing Modes

Addressing in the CPU32L is register-oriented. Most instructions allow the results of the specified operation to be placed either in a register or directly in memory. This flexibility eliminates the need for extra instructions to store register contents in memory. The CPU32L supports seven basic addressing modes:

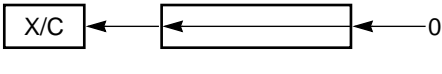
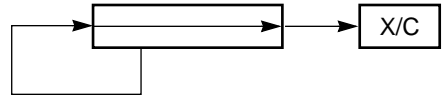
- Register direct
- Register indirect
- Register indirect with index
- Program counter indirect with displacement
- Program counter indirect with index
- Absolute
- Immediate

Included in the register indirect addressing modes are the capabilities to post-increment, predecrement, and offset. The program counter relative mode also has index and offset capabilities. In addition to these addressing modes, many instructions implicitly specify the use of the status register, stack pointer, or program counter.

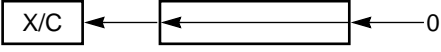
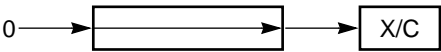
#### 4.6 Instruction Set Summary

**Table 33** provides a summary of the CPU32L instruction set.

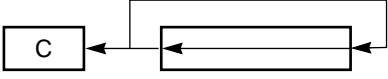
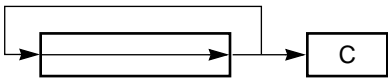
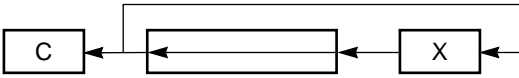
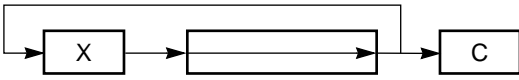
**Table 33 Instruction Set Summary**

Instruction	Syntax	Operand Size	Operation
ABCD	Dn, Dn – (An), – (An)	8 8	Source <sub>10</sub> + Destination <sub>10</sub> + X ⇒ Destination
ADD	Dn, <ea> <ea>, Dn	8, 16, 32 8, 16, 32	Source + Destination ⇒ Destination
ADDA	<ea>, An	16, 32	Source + Destination ⇒ Destination
ADDI	#<data>, <ea>	8, 16, 32	Immediate data + Destination ⇒ Destination
ADDQ	# <data>, <ea>	8, 16, 32	Immediate data + Destination ⇒ Destination
ADDX	Dn, Dn – (An), – (An)	8, 16, 32 8, 16, 32	Source + Destination + X ⇒ Destination
AND	<ea>, Dn Dn, <ea>	8, 16, 32 8, 16, 32	Source • Destination ⇒ Destination
ANDI	# <data>, <ea>	8, 16, 32	Data • Destination ⇒ Destination
ANDI to CCR	# <data>, CCR	8	Source • CCR ⇒ CCR
ANDI to SR1 <sup>1</sup>	# <data>, SR	16	Source • SR ⇒ SR
ASL	Dn, Dn # <data>, Dn <ea>	8, 16, 32 8, 16, 32 16	
ASR	Dn, Dn # <data>, Dn <ea>	8, 16, 32 8, 16, 32 16	
Bcc	label	8, 16, 32	If condition true, then PC + d ⇒ PC
BCHG	Dn, <ea> # <data>, <ea>	8, 32 8, 32	$\overline{((\text{bit number}) \text{ of destination})} \Rightarrow Z \Rightarrow \text{bit of destination}$
BCLR	Dn, <ea> # <data>, <ea>	8, 32 8, 32	$\overline{((\text{bit number}) \text{ of destination})} \Rightarrow Z;$ 0 ⇒ bit of destination
BGND	none	none	If background mode enabled, then enter background mode, else format/vector ⇒ – (SSP); PC ⇒ – (SSP); SR ⇒ – (SSP); (vector) ⇒ PC
BKPT	# <data>	none	If breakpoint cycle acknowledged, then execute returned operation word, else trap as illegal instruction
BRA	label	8, 16, 32	PC + d ⇒ PC
BSET	Dn, <ea> # <data>, <ea>	8, 32 8, 32	$\overline{((\text{bit number}) \text{ of destination})} \Rightarrow Z;$ 1 ⇒ bit of destination
BSR	label	8, 16, 32	SP – 4 ⇒ SP; PC ⇒ (SP); PC + d ⇒ PC
BTST	Dn, <ea> # <data>, <ea>	8, 32 8, 32	$\overline{((\text{bit number}) \text{ of destination})} \Rightarrow Z$
CHK	<ea>, Dn	16, 32	If Dn < 0 or Dn > (ea), then CHK exception
CHK2	<ea>, Rn	8, 16, 32	If Rn < lower bound or Rn > upper bound, then CHK exception
CLR	<ea>	8, 16, 32	0 ⇒ Destination
CMP	<ea>, Dn	8, 16, 32	(Destination – Source), CCR shows results
CMPA	<ea>, An	16, 32	(Destination – Source), CCR shows results
CMPI	# <data>, <ea>	8, 16, 32	(Destination – Data), CCR shows results

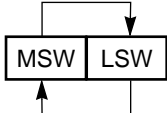
**Table 33 Instruction Set Summary (Continued)**

Instruction	Syntax	Operand Size	Operation
CMPM	(An) +, (An) +	8, 16, 32	(Destination – Source), CCR shows results
CMP2	<ea>, Rn	8, 16, 32	Lower bound ≤ Rn ≤ Upper bound, CCR shows result
DBcc	Dn, label	16	If condition false, then Dn – 1 ⇒ PC; if Dn ≠ (– 1), then PC + d ⇒ PC
DIVS/DIVU	<ea>, Dn	32/16 ⇒ 16 : 16	Destination / Source ⇒ Destination (signed or unsigned)
DIVSL/DIVUL	<ea>, Dr : Dq <ea>, Dq <ea>, Dr : Dq	64/32 ⇒ 32 : 32 32/32 ⇒ 32 32/32 ⇒ 32 : 32	Destination / Source ⇒ Destination (signed or unsigned)
EOR	Dn, <ea>	8, 16, 32	Source ⊕ Destination ⇒ Destination
EORI	# <data>, <ea>	8, 16, 32	Data ⊕ Destination ⇒ Destination
EORI to CCR	# <data>, CCR	8	Source ⊕ CCR ⇒ CCR
EORI to SR <sup>1</sup>	# <data>, SR	16	Source ⊕ SR ⇒ SR
EXG	Rn, Rn	32	Rn ⇒ Rn
EXT	Dn Dn	8 ⇒ 16 16 ⇒ 32	Sign extended Destination ⇒ Destination
EXTB	Dn	8 ⇒ 32	Sign extended Destination ⇒ Destination
ILLEGAL	none	none	SSP – 2 ⇒ SSP; vector offset ⇒ (SSP); SSP – 4 ⇒ SSP; PC ⇒ (SSP); SSP – 2 ⇒ SSP; SR ⇒ (SSP); Illegal instruction vector address ⇒ PC
JMP	<ea>	none	Destination ⇒ PC
JSR	<ea>	none	SP – 4 ⇒ SP; PC ⇒ (SP); destination ⇒ PC
LEA	<ea>, An	32	<ea> ⇒ An
LINK	An, # d	16, 32	SP – 4 ⇒ SP, An ⇒ (SP); SP ⇒ An, SP + d ⇒ SP
LPSTOP <sup>1, 2</sup>	# <data>	16	Data ⇒ SR; interrupt mask ⇒ EBI; STOP
LSL	Dn, Dn # <data>, Dn <ea>	8, 16, 32 8, 16, 32 16	
LSR	Dn, Dn # <data>, Dn <ea>	8, 16, 32 8, 16, 32 16	
MOVE	<ea>, <ea>	8, 16, 32	Source ⇒ Destination
MOVEA	<ea>, An	16, 32 ⇒ 32	Source ⇒ Destination
MOVEA <sup>1</sup>	USP, An An, USP	32 32	USP ⇒ An An ⇒ USP
MOVE from CCR	CCR, <ea>	16	CCR ⇒ Destination
MOVE to CCR	<ea>, CCR	16	Source ⇒ CCR
MOVE from SR <sup>1</sup>	SR, <ea>	16	SR ⇒ Destination
MOVE to SR <sup>1</sup>	<ea>, SR	16	Source ⇒ SR
MOVE USP <sup>1</sup>	USP, An An, USP	32 32	USP ⇒ An An ⇒ USP
MOVEC <sup>1</sup>	Rc, Rn Rn, Rc	32 32	Rc ⇒ Rn Rn ⇒ Rc

**Table 33 Instruction Set Summary (Continued)**

Instruction	Syntax	Operand Size	Operation
MOVEM	list, <ea> <ea>, list	16, 32 16, 32 ⇒ 32	Listed registers ⇒ Destination Source ⇒ Listed registers
MOVEP	Dn, (d16, An)  (d16, An), Dn	16, 32	Dn [31 : 24] ⇒ (An + d); Dn [23 : 16] ⇒ (An + d + 2); Dn [15 : 8] ⇒ (An + d + 4); Dn [7 : 0] ⇒ (An + d + 6)  (An + d) ⇒ Dn [31 : 24]; (An + d + 2) ⇒ Dn [23 : 16]; (An + d + 4) ⇒ Dn [15 : 8]; (An + d + 6) ⇒ Dn [7 : 0]
MOVEQ	#<data>, Dn	8 ⇒ 32	Immediate data ⇒ Destination
MOVES <sup>1</sup>	Rn, <ea> <ea>, Rn	8, 16, 32	Rn ⇒ Destination using DFC Source using SFC ⇒ Rn
MULS/MULU	<ea>, Dn <ea>, D1 <ea>, Dh : D1	16 * 16 ⇒ 32 32 * 32 ⇒ 32 32 * 32 ⇒ 64	Source * Destination ⇒ Destination (signed or unsigned)
NBCD	<ea>	8 8	0 – Destination <sub>10</sub> – X ⇒ Destination
NEG	<ea>	8, 16, 32	0 – Destination ⇒ Destination
NEGX	<ea>	8, 16, 32	0 – Destination – X ⇒ Destination
NOP	none	none	PC + 2 ⇒ PC
NOT	<ea>	8, 16, 32	$\overline{\text{Destination}} \Rightarrow \text{Destination}$
OR	<ea>, Dn Dn, <ea>	8, 16, 32 8, 16, 32	Source + Destination ⇒ Destination
ORI	#<data>, <ea>	8, 16, 32	Data + Destination ⇒ Destination
ORI to CCR	#<data>, CCR	16	Source + CCR ⇒ SR
ORI to SR <sup>1</sup>	#<data>, SR	16	Source ; SR ⇒ SR
PEA	<ea>	32	SP – 4 ⇒ SP; <ea> ⇒ SP
RESET <sup>1</sup>	none	none	Assert $\overline{\text{RESET}}$ line
ROL	Dn, Dn #<data>, Dn <ea>	8, 16, 32 8, 16, 32 16	
ROR	Dn, Dn #<data>, Dn <ea>	8, 16, 32 8, 16, 32 16	
ROXL	Dn, Dn #<data>, Dn <ea>	8, 16, 32 8, 16, 32 16	
ROXR	Dn, Dn #<data>, Dn <ea>	8, 16, 32 8, 16, 32 16	
RTD	#d	16	(SP) ⇒ PC; SP + 4 + d ⇒ SP
RTE <sup>1</sup>	none	none	(SP) ⇒ SR; SP + 2 ⇒ SP; (SP) ⇒ PC; SP + 4 ⇒ SP; Restore stack according to format
RTR	none	none	(SP) ⇒ CCR; SP + 2 ⇒ SP; (SP) ⇒ PC; SP + 4 ⇒ SP
RTS	none	none	(SP) ⇒ PC; SP + 4 ⇒ SP
SBCD	Dn, Dn – (An), – (An)	8 8	Destination <sub>10</sub> – Source <sub>10</sub> – X ⇒ Destination

**Table 33 Instruction Set Summary (Continued)**

Instruction	Syntax	Operand Size	Operation
Scc	<ea>	8	If condition true, then destination bits are set to 1; else, destination bits are cleared to 0
STOP <sup>1</sup>	#<data>	16	Data ⇒ SR; STOP
SUB	<ea>, Dn Dn, <ea>	8, 16, 32	Destination – Source ⇒ Destination
SUBA	<ea>, An	16, 32	Destination – Source ⇒ Destination
SUBI	#<data>, <ea>	8, 16, 32	Destination – Data ⇒ Destination
SUBQ	#<data>, <ea>	8, 16, 32	Destination – Data ⇒ Destination
SUBX	Dn, Dn – (An), – (An)	8, 16, 32 8, 16, 32	Destination – Source – X ⇒ Destination
SWAP	Dn	16	
TAS	<ea>	8	Destination Tested Condition Codes bit 7 of Destination
TBLS/TBLU	<ea>, Dn Dym : Dyn, Dn	8, 16, 32	Dyn – Dym ⇒ Temp (Temp * Dn [7 : 0]) ⇒ Temp (Dym * 256) + Temp ⇒ Dn
TBLSN/TBLUN	<ea>, Dn Dym : Dyn, Dn	8, 16, 32	Dyn – Dym ⇒ Temp (Temp * Dn [7 : 0]) / 256 ⇒ Temp Dym + Temp ⇒ Dn
TRAP	#<data>	none	SSP – 2 ⇒ SSP; format/vector offset ⇒ (SSP); SSP – 4 ⇒ SSP; PC ⇒ (SSP); SR ⇒ (SSP); vector address ⇒ PC
TRAPcc	none #<data>	none 16, 32	If cc true, then TRAP exception
TRAPV	none	none	If V set, then overflow TRAP exception
TST	<ea>	8, 16, 32	Source – 0, to set condition codes
UNLK	An	32	An ⇒ SP; (SP) ⇒ An, SP + 4 ⇒ SP

1. Privileged instruction.

2. Two LPSTOP modes are supported. The first LPSTOP mode is the normal LPSTOP in which the system clock on the chip is stopped, shutting down all IMB modules with the exception of some parts of the SIML and CLKOUT. The second LPSTOP mode causes the system clock to be stopped only at the CPU32L, when the LPSTOP instruction is executed by the CPU32L. As with the normal LPSTOP operation, the CPU32L can be restarted by an interrupt, a trace, or a reset exception.

## 4.7 Background Debugging Mode

The background debugger on the CPU32L is implemented in CPU microcode. The background debugging commands are summarized in **Table 34**.

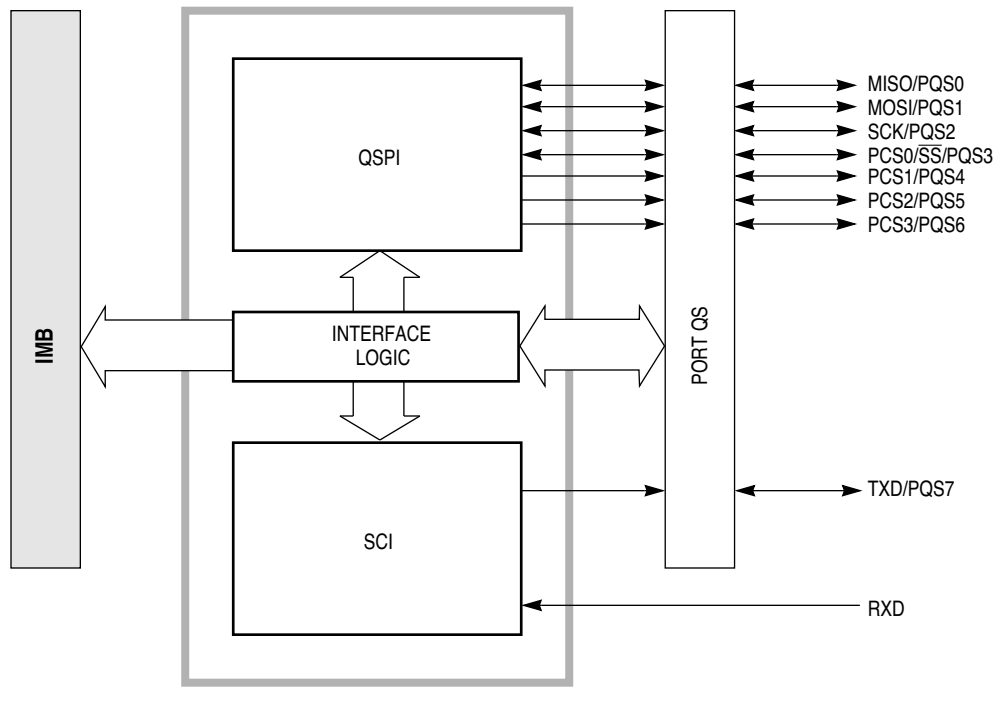
**Table 34 Background Debugging Mode Commands**

Command	Mnemonic	Description
Read D/A Register	RDREG/RAREG	Read the selected address or data register and return the results through the serial interface.
Write D/A Register	WDREG/WAREG	The data operand is written to the specified address or data register.
Read System Register	RSREG	The specified system control register is read. All registers that can be read in supervisor mode can be read in background mode.
Write System Register	WSREG	The operand data is written into the specified system control register.
Read Memory Location	READ	Read the sized data at the memory location specified by the long-word address. The source function code register (SFC) determines the address space accessed.
Write Memory Location	WRITE	Write the operand data to the memory location specified by the long-word address. The destination function code (DFC) register determines the address space accessed.
Dump Memory Block	DUMP	Used in conjunction with the READ command to dump large blocks of memory. An initial READ is executed to set up the starting address of the block and retrieve the first result. Subsequent operands are retrieved with the DUMP command.
Fill Memory Block	FILL	Used in conjunction with the WRITE command to fill large blocks of memory. Initially, a WRITE is executed to set up the starting address of the block and supply the first operand. The FILL command writes subsequent operands.
Resume Execution	GO	The pipe is flushed and refilled before resuming instruction execution at the current PC.
Patch User Code	CALL	Current program counter is stacked at the location of the current stack pointer. Instruction execution begins at user patch code.
Reset Peripherals	RST	Asserts RESET for 512 clock cycles. The CPU is not reset by this command. Synonymous with the CPU RESET instruction.
No Operation	NOP	NOP performs no operation and can be used as a null command.



## 5 Queued Serial Module

The QSM contains two serial interfaces, the queued serial peripheral interface (QSPI) and the serial communication interface (SCI). **Figure 14** shows the QSM block diagram.



**Figure 14 QSM Block Diagram**

### 5.1 Overview

The QSPI provides easy peripheral expansion or interprocessor communication through a full-duplex, synchronous, three-line bus: data in, data out, and a serial clock. Four programmable peripheral chip-select pins provide addressability for up to 16 peripheral devices. A self-contained RAM queue allows up to 16 serial transfers of 8 to 16 bits each, or transmission of a 256-bit data stream without CPU intervention. A special wraparound mode supports continuous sampling of a serial peripheral, with automatic QSPI RAM updating, which makes the interface to A/D converters more efficient.

The SCI provides a standard non-return to zero (NRZ) mark/space format. It operates in either full- or half-duplex mode. There are separate transmitter and receiver enable bits and dual data buffers. A modulus-type baud rate generator provides rates from 55 to 451 kbaud with a 14.44-MHz system clock. Word length of either eight or nine bits is software selectable. Optional parity generation and detection provide either even or odd parity check capability. Advanced error detection circuitry catches glitches of up to 1/16 of a bit time in duration. Wakeup functions allow the CPU to run uninterrupted until meaningful data is available.

**Table 35** shows the address map of the QSM.

## 5.2 Address Map

The “Access” column in the QSM address map in **Table 35** indicates which registers are accessible only at the supervisor privilege level and which can be assigned to either the supervisor or user privilege level, according to the value of the SUPV bit in the QSMCR.

**Table 35 QSM Address Map**

Access	Address	15	8	7	0
S	\$YFFC00 <sup>1</sup>	QSM Module Configuration Register (QSMCR)			
S	\$YFFC02	QSM Test Register (QTEST)			
S	\$YFFC04	QSM Interrupt Level Register (QILR)		QSM Interrupt Vector Register (QIVR)	
S/U	\$YFFC06	Not Used			
S/U	\$YFFC08	SCI Control 0 Register (SCCR0)			
S/U	\$YFFC0A	SCI Control 1 Register (SCCR1)			
S/U	\$YFFC0C	SCI Status Register (SCSR)			
S/U	\$YFFC0E	SCI Data Register (SCDR)			
S/U	\$YFFC10	Not Used			
S/U	\$YFFC12	Not Used			
S/U	\$YFFC14	Not Used		PQS Data Register (PORTQS)	
S/U	\$YFFC16	PQS Pin Assignment Register (PQSPAR)		PQS Data Direction Register (DDRQS)	
S/U	\$YFFC18	SPI Control Register 0 (SPCR0)			
S/U	\$YFFC1A	SPI Control Register 1 (SPCR1)			
S/U	\$YFFC1C	SPI Control Register 2 (SPCR2)			
S/U	\$YFFC1E	SPI Control Register 3 (SPCR3)		SPI Status Register (SPSR)	
S/U	\$YFFC20 – \$YFFCFF	Not Used			
S/U	\$YFFD00 – \$YFFD1F	Receive RAM (RR[0:F])			
S/U	\$YFFD20 – \$YFFD3F	Transmit RAM (TR[0:F])			
S/U	\$YFFD40 – \$YFFD4F	Command RAM (CR[0:F])			

**NOTES:**

1. Y = M111, where M is the logic state of the MM bit in the SIMLCR.

### 5.3 Pin Function

**Table 36** is a summary of the functions of the QSM pins when they are not configured for general-purpose I/O. The QSM data direction register (DDRQS) designates each pin except RXD as an input or output.

**Table 36 QSM Pin Functions**

	Pin	Mode	Pin Function
QSPI Pins	MISO	Master	Serial data input to QSPI
		Slave	Serial data output from QSPI
	MOSI	Master	Serial data output from QSPI
		Slave	Serial data input to QSPI
	SCK	Master	Clock output from QSPI
		Slave	Clock input to QSPI
PCS0/ $\overline{SS}$	Master	Input: Assertion causes mode fault Output: Selects peripherals	
	Slave	Input: Selects the QSPI	
SCI Pins	PCS[3:1]	Master	Output: Selects peripherals
		Slave	None
	TXD	Transmit	Serial data output from SCI
	RXD	Receive	Serial data input to SCI

### 5.4 QSM Registers

QSM registers 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 separate sections below. Writes to unimplemented register bits have no meaning or effect, and reads from unimplemented bits always return a logic zero value.

The module mapping bit of the SIML configuration register (SIMLCR) defines the most significant bit (ADDR23) of the address, shown in each register figure as Y (Y = \$7 or \$F). This bit, concatenated with the rest of the address given, forms the absolute address of each register. Refer to the SIML section of this technical summary for more information about how the state of MM affects the system.

#### 5.4.1 Global Registers

The QSM global registers contain system parameters used by both the QSPI and the SCI submodules. These registers contain the bits and fields used to configure the QSM.

#### QSMCR — QSM Configuration Register

**\$YFFC00**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP	FRZ1	FRZ0	0	0	0	0	0	SUPV	0	0	0	IARB			

RESET:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

The QSMCR contains parameters for the QSM/CPU/intermodule bus (IMB) interface.

### STOP — Stop Enable

- 0 = Normal QSM clock operation
- 1 = QSM clock operation stopped

STOP places the QSM in a low-power state by disabling the system clock in most parts of the module. The 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 to be valid while STOP is asserted. STOP can be negated by the CPU and by reset.

The system software must stop each submodule before asserting STOP to avoid complications at re-start 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.

### FRZ1 — Freeze 1

- 0 = Ignore the FREEZE signal on the IMB
- 1 = Halt the QSPI (on a transfer boundary)

FRZ1 determines what action is taken by the QSPI when the FREEZE signal of the IMB is asserted. FREEZE is asserted whenever the CPU enters the background mode.

### FRZ0 — Freeze 0

Reserved

### Bits [12:8] — Not Implemented

### SUPV — Supervisor/Unrestricted

- 0 = User access
- 1 = Supervisor access

SUPV defines the assignable QSM registers as either supervisor-only data space or unrestricted data space.

### IARB — Interrupt Arbitration Identification Number

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. Refer to **3.9 Interrupts** for more information.

### QTEST — QSM Test Register

**\$YFFC02**

QTEST is used during factory testing of the QSM. Accesses to QTEST must be made while the MCU is in test mode.

### QILR — QSM Interrupt Levels Register

**\$YFFC04**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	ILQSPI			ILSCI			QIVR							

RESET:

0 0 0 0 0 0 0 0

QILR determines the priority level of interrupts requested by the QSM and the vector used when an interrupt is acknowledged.

### ILQSPI — Interrupt Level for QSPI

ILQSPI determines the priority of QSPI interrupts. This field must be given a value between \$0 (interrupts disabled) to \$7 (highest priority).

### ILSCI — Interrupt Level of SCI

ILSCI determines the priority of SCI interrupts. This field must be given a value between \$0 (interrupts disabled) to \$7 (highest priority).

If ILQSPI and ILSCI are the same nonzero value, and both submodules simultaneously request interrupt service, QSPI has priority.

### QIVR — QSM Interrupt Vector Register

**\$YFFC05**

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

At reset, QIVR is initialized to \$0F, which corresponds to the uninitialized interrupt vector in the exception table. This vector is selected until QIVR is written. A user-defined vector (\$40–\$FF) should be written to QIVR during QSM initialization.

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 interrupt-acknowledge cycle is supplied by the QSM. During an interrupt-acknowledge cycle, INTV[7:1] are driven on DATA[7:1] IMB lines. DATA0 is negated for an SCI interrupt and asserted for a QSPI interrupt. Writes to INTV0 have no meaning or effect. Reads of INTV0 return a value of one.

### 5.4.2 Pin Control Registers

The QSM uses nine pins, eight of which form a parallel port (PORTQS) on the MCU. Although these pins are used by the serial subsystems, any pin can alternately be assigned as general-purpose I/O on a pin-by-pin basis.

Pins used for general-purpose I/O must not be assigned to the QSPI by register PQSPAR. To avoid driving incorrect data, the first byte to be output must be written before DDRQS is configured. DDRQS must then be written to determine the direction of data flow and to output the value contained in register PORTQS. Subsequent data for output is written to PORTQS.

### PORTQS — Port QS Data Register

**\$YFFC14**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								PQS7	PQS6	PQS5	PQS4	PQS3	PQS2	PQS1	PQS0
RESET:															
								0	0	0	0	0	0	0	0

PORTQS latches I/O data. Writes drive pins defined as outputs. Reads return data present on the pins. To avoid driving undefined data, first write a byte to PORTQS, then configure DDRQS.

**PQSPAR** — PORT QS Pin Assignment Register  
**DDRQS** — PORT QS Data Direction Register

**\$YFFC16**  
**\$YFFC17**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	PQSPA6	PQSPA5	PQSPA4	PQSPA3	0	PQSPA1	PQSPA0	DDQS7	DDQS6	DDQS5	DDQS4	DDQS3	DDQS2	DDQS1	DDQS0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Clearing a bit in the PQSPAR assigns the corresponding pin to general-purpose I/O; setting a bit assigns the pin to the QSPI. The PQSPAR does not affect operation of the SCI. **Table 37** displays PQSPAR pin assignments.

**Table 37 PQSPAR Pin Assignments**

PQSPAR Field	PQSPAR Bit	Pin Function
PQSPA0	0	PQS0
	1	MISO
PQSPA1	0	PQS1
	1	MOSI
PQSPA2	0	PQS2 <sup>1</sup>
	1	SCK
PQSPA3	0	PQS3
	1	PCS0/ $\overline{SS}$
PQSPA4	0	PQS4
	1	PCS1
PQSPA5	0	PQS5
	1	PCS2
PQSPA6	0	PQS6
	1	PCS3
PQSPA7	0	PQS7 <sup>2</sup>
	1	TXD

NOTES:

1. PQS2 is a digital I/O pin unless the SPI is enabled (SPE in SPCR1 set), in which case it becomes SPI serial clock SCK.
2. PQS7 is a digital I/O pin unless the SCI transmitter is enabled (TE in SCCR1 = 1), in which case it becomes SCI serial output TXD.

DDRQS determines whether pins are inputs or outputs. Clearing a bit makes the corresponding pin an input; setting a bit makes the pin an output. DDRQS affects both QSPI function and I/O function. **Table 38** shows the effect of DDRQS on QSM pin functions.

**Table 38 Effect of DDRQS on QSM Pin Function**

QSM Pin	Mode	DDRQS Bit	Bit State	Pin Function
MISO	Master	DDQ0	0	Serial data input to QSPI
			1	Disables data input
	Slave		0	Disables data output
			1	Serial data output from QSPI
MOSI	Master	DDQ1	0	Disables data output
			1	Serial data output from QSPI
	Slave		0	Serial data input to QSPI
			1	Disables data input
SCK <sup>1</sup>	Master	DDQ2	0	Disables clock output
			1	Clock output from QSPI
	Slave		0	Clock input to QSPI
			1	Disables clock Input
PCS0/ $\overline{SS}$	Master	DDQ3	0	Assertion causes mode fault
			1	Chip-select output
	Slave		0	QSPI slave select input
			1	Disables select input
PCS[3:1]	Master	DDQ[4:6]	0	Disables chip-select output
			1	Chip-select output
	Slave		0	Inactive
			1	Inactive
TXD <sup>2</sup>	Transmit	DDQ7	X	Serial data output from SCI
RXD	Receive	None	NA	Serial data input to SCI

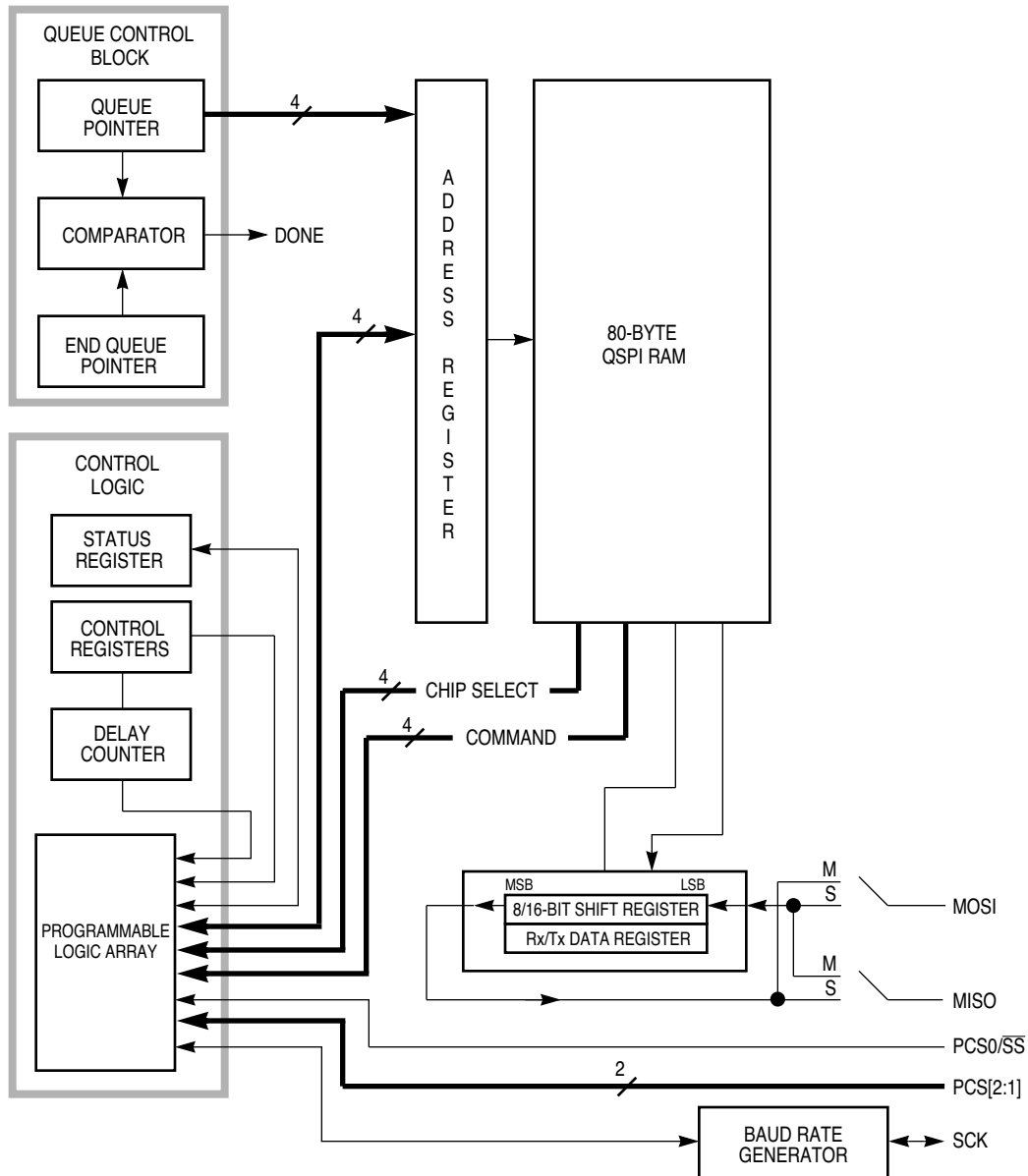
NOTES:

1. PQS2 is a digital I/O pin unless the SPI is enabled (SPE in SPCR1 set), in which case it becomes SPI serial clock SCK.
2. PQS7 is a digital I/O pin unless the SCI transmitter is enabled (TE in SCCR1 = 1), in which case it becomes SCI serial output TXD.

DDRQS determines the direction of the TXD pin only when the SCI transmitter is disabled. When the SCI transmitter is enabled, the TXD pin is an output.

## 5.5 QSPI Submodule

The QSPI submodule communicates with external devices through a synchronous serial bus. The QSPI is fully compatible with the serial peripheral interface (SPI) systems found on other Motorola products. **Figure 15** shows a block diagram of the QSPI.



QSPI BLOCK

Figure 15 QSPI Block Diagram



### 5.5.1 QSPI Pins

Seven pins are associated with the QSPI. When not needed for a QSPI function, they can be configured as general-purpose I/O pins. The PCS0/ $\overline{SS}$  pin can function as a peripheral chip select output, slave select input, or general-purpose I/O. Refer to **Table 39** for QSPI input and output pins and their functions.

**Table 39 QSPI Pins**

Pin Name(s)	Mnemonic(s)	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	Master Slave	Clock output from QSPI Clock input to QSPI
Peripheral Chip Selects	PCS[3:1]	Master	Select peripherals
Peripheral Chip Select Slave Select	PCS0 $\overline{SS}$	Master Master Slave	Selects peripheral Causes mode fault Initiates serial transfer

### 5.5.2 QSPI Registers

The programmer's model for the QSPI submodule consists of the QSM global and pin control registers, four QSPI control registers, one status register, and the 80-byte QSPI RAM.

The CPU can read and write to registers and RAM. The four control registers must be initialized before the QSPI is enabled to ensure defined operation. SPCR1 should be written last because it contains QSPI enable bit SPE. Asserting this bit starts the QSPI. The QSPI control registers are reset to a defined state and can then be changed by the CPU. Reset values are shown below each register.

**Table 40** shows a memory map of the QSPI.

**Table 40 QSPI Memory Map**

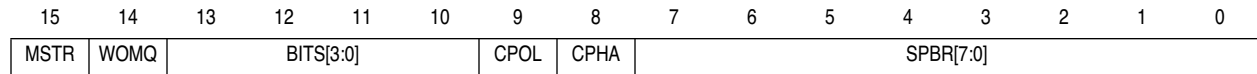
Address	Name	Usage
\$YFFC18	SPCR0	QSPI control register 0
\$YFFC1A	SPCR1	QSPI control register 1
\$YFFC1C	SPCR2	QSPI control register 2
\$YFFC1E	SPCR3	QSPI control register 3
\$YFFC1F	SPSR	QSPI status register
\$YFFD00	RR[0:F]	QSPI receive data (16 words)
\$YFFD20	TR[0:F]	QSPI transmit data (16 words)
\$YFFD40	CR[0:F]	QSPI command control (8 words)

Writing a different value into any control register except SPCR2 while the QSPI is enabled disrupts operation. SPCR2 is buffered to prevent disruption of the current serial transfer. After completion of the current serial transfer, the new SPCR2 values become effective.

Writing the same value into any control register except SPCR2 while the QSPI is enabled has no effect on QSPI operation. Rewriting NEWQP[3:0] in SPCR2 causes execution to restart at the designated location.

**SPCR0** — QSPI Control Register 0

**\$YFFC18**



RESET:

0    0    0    0    0    0    0    1    0    0    0    0    0    1    0    0

SPCR0 contains parameters for configuring the QSPI before it is enabled. The CPU can read and write this register. The QSM has read-only access.

**MSTR** — Master/Slave Mode Select

- 0 = QSPI is a slave device and only responds to externally generated serial data.
- 1 = QSPI is system master and can initiate transmission to external SPI devices.

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.

**WOMQ** — Wired-OR Mode for QSPI Pins

- 0 = Outputs have normal MOS drivers.
- 1 = Pins designated for output by DDRQS have open-drain drivers.

WOMQ allows the wired-OR function to be used on QSPI pins, regardless of whether they are used as general-purpose outputs or as QSPI outputs. WOMQ affects the QSPI pins regardless of whether the QSPI is enabled or disabled.

**BITS[3:0]** — Bits Per Transfer

In master mode, when BITSE in a command is set, the BITS[3:0] field determines the number of data bits transferred. When BITSE is cleared, eight bits are transferred. Reserved values default to eight bits. In slave mode, the command RAM is not used and the setting of BITSE has no effect on QSPI transfers. Instead, the BITS[3:0] field determines the number of bits the QSPI will receive during each transfer before storing the received data. **Table 41** shows the number of bits per transfer.

**Table 41 Bits Per Transfer**

BITS[3:0]	Bits Per Transfer
0000	16
0001	Reserved
0010	Reserved
0011	Reserved
0100	Reserved
0101	Reserved
0110	Reserved
0111	Reserved
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

### CPOL — Clock Polarity

0 = The inactive state value of SCK is logic level zero.

1 = The inactive state value of SCK is logic level one.

CPOL is used to determine the inactive state value of the serial clock (SCK). It is used with CPHA to produce a desired clock/data relationship between master and slave devices.

### CPHA — Clock Phase

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.

CPHA determines which edge of SCK causes data to change and which edge causes data to be captured. CPHA is used with CPOL to produce a desired clock/data relationship between master and slave devices. CPHA is set at reset.

### SPBR[7:0] — Serial Clock Baud Rate

The QSPI uses a modulus counter to derive SCK baud rate from the MCU system clock. Baud rate is selected by writing a value from 2 to 255 into the SPBR[7:0] field. The following equation determines the SCK baud rate:

$$\text{SCK Baud Rate} = \frac{\text{System Clock}}{2 \times \text{SPBR}[7:0]}$$

or

$$\text{SPBR}[7:0] = \frac{\text{System Clock}}{2 \times \text{SCK Baud Rate Desired}}$$

Giving SPBR[7:0] a value of zero or one disables the baud rate generator. SCK is disabled and assumes its inactive state value. No serial transfers occur. At reset, baud rate is initialized to one eighth of the system clock frequency.

### SPRC1 — QSPI Control Register 1

**\$YFFC1A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPE	DSCKL[6:0]						DTL[7:0]								

RESET:

0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0

SPCR1 contains parameters for configuring the QSPI before it is enabled. The CPU can read and write this register, but the QSM has read access only, except for SPE, which is automatically cleared by the QSPI after completing all serial transfers, or when a mode fault occurs.

### SPE — QSPI Enable

0 = QSPI is disabled. QSPI pins can be used for general-purpose I/O.

1 = QSPI is enabled. Pins allocated by PQSPAR are controlled by the QSPI.

### DSCKL[6:0] — Delay before SCK

When the DSCK bit in command RAM is set, this field determines the length of delay from PCS valid to SCK transition. PCS can be any of the four peripheral chip-select pins. The following equation determines the actual delay before SCK:

$$\text{PCS to SCK Delay} = \frac{\text{DSCKL}[6:0]}{\text{System Clock}}$$

where DSCKL[6:0] equals {1, 2, 3, ..., 127}.

When the DSCK value of a queue entry equals zero, then DSCKL[6:0] is not used. Instead, the PCS valid-to-SCK transition is one-half SCK period.

### DTL[7:0] — Length of Delay after Transfer

When the DT bit in command RAM is set, this field determines the length of delay after serial transfer. The following equation is used to calculate the delay:

$$\text{Delay after Transfer} = \frac{32 \times \text{DTL}[7:0]}{\text{System Clock}}$$

where DTL equals {1, 2, 3,..., 255}.

A zero value for DTL[7:0] causes a delay-after-transfer value of 8192/System Clock.

If DT equals zero, a standard delay is inserted.

$$\text{Standard Delay after Transfer} = \frac{17}{\text{System Clock}}$$

Delay after transfer can be used to provide a peripheral deselect interval. A delay can also be inserted between consecutive transfers to allow serial A/D converters to complete conversion.

### SPCR2 — QSPI Control Register 2

**\$YFFC1C**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPIFIE	WREN	WRTO	0	ENDQP[3:0]			0	0	0	0	NEWQP[3:0]				

RESET:

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

SPCR2 contains QSPI configuration parameters. The CPU can read and write this register; the QSM has read access only. Writes to SPCR2 are buffered. A write to SPCR2 that changes a bit value while the QSPI is operating is ineffective on the current serial transfer, but becomes effective on the next serial transfer. Reads of SPCR2 return the current value of the register, not of the buffer.

#### SPIFIE — SPI Finished Interrupt Enable

0 = QSPI interrupts disabled

1 = QSPI interrupts enabled

SPIFIE enables the QSPI to generate a CPU interrupt upon assertion of the status flag SPIF.

#### WREN — Wrap Enable

0 = Wraparound mode disabled

1 = Wraparound mode enabled

WREN enables or disables wraparound mode.

#### WRTO — Wrap To

When wraparound mode is enabled, after the end of queue has been reached, WRTO determines which address the QSPI executes.

#### Bit 12 — Not Implemented

#### ENDQP[3:0] — Ending Queue Pointer

This field contains the last QSPI queue address.

#### Bits [7:4] — Not Implemented

#### NEWQP[3:0] — New Queue Pointer Value

This field contains the first QSPI queue address.

**SPCR3 — QSPI Control Register****\$YFFC1E**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	LOOPQ	HMIE	HALT	SPSR							

RESET:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

SPCR3 contains QSPI configuration parameters. The CPU can read and write SPCR3, but the QSM has read-only access.

Bits [15:11] — Not Implemented

**LOOPQ — QSPI Loop Mode**

0 = Feedback path disabled

1 = Feedback path enabled

LOOPQ controls feedback on the data serializer for testing.

**HMIE — HALTA and MODF Interrupt Enable**

0 = HALTA and MODF interrupts disabled

1 = HALTA and MODF interrupts enabled

HMIE controls CPU interrupts caused by the HALTA status flag or the MODF status flag in SPSR.

**HALT — Halt**

0 = Halt not enabled

1 = Halt enabled

When HALT is asserted, the QSPI stops on a queue boundary. It is in a defined state from which it can later be restarted.

**SPSR — QSPI Status Register****\$YFFC1F**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPCR3							SPIF	MODF	HALTA	0	CPTQP[3:0]				

RESET:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

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 it to clear status flags.

**SPIF — QSPI Finished Flag**

0 = QSPI not finished

1 = QSPI finished

SPIF is set after execution of the command at the address in ENDQP[3:0].

**MODF — Mode Fault Flag**

0 = Normal operation

1 = Another SPI node requested to become the network SPI master while the QSPI was enabled in master mode ( $\overline{SS}$  input taken low).

The QSPI asserts MODF when the QSPI is the serial master (MSTR = 1) and the  $\overline{SS}$  input pin is negated by an external driver.

**HALTA — Halt Acknowledge Flag**

0 = QSPI not halted

1 = QSPI halted

HALTA is asserted when the QSPI halts in response to CPU assertion of HALT.

Bit 4 — Not Implemented

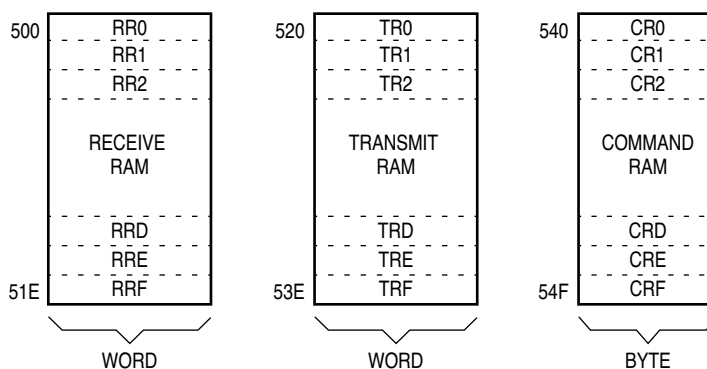
### CPTQP[3:0] — Completed Queue Pointer

CPTQP[3:0] points to the last command executed. It is updated when the current command is complete. When the first command in a queue is executing, CPTQP[3:0] contains either the reset value (\$0) or a pointer to the last command completed in the previous queue.

### 5.5.3 QSPI RAM

The QSPI contains an 80-byte block of dual-access static RAM that is used by both the QSPI and the CPU. The RAM is divided into three segments: receive data, transmit data, and command control data. 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. Command control data is used to perform the transfer.

Figure 16 displays the organization of the RAM.



QSPI RAM MAP

Figure 16 QSPI RAM

Once the CPU has set up the queue of QSPI commands and enabled the QSPI, the QSPI can operate 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. It is possible to execute a queue of commands repeatedly without CPU intervention.

### RR[0:F] — Receive Data RAM

**\$YFFD00**

Data received by the QSPI is stored in this segment. The CPU reads this segment to retrieve data from the QSPI. Data stored in receive RAM is right-justified. 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[3:0] 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.

### TR[0:F] — Transmit Data RAM

**\$YFFD20**

Data that is to be transmitted by the QSPI is stored in this segment. The CPU usually writes one word of data into this segment for each queue command to be executed.

Information to be transmitted must be written to transmit data RAM in a right-justified format. The QSPI cannot modify information in the transmit data RAM. The QSPI copies the information to its data serializer for transmission. Information remains in transmit RAM until overwritten.

**CR[0:F] — Command RAM**

**\$YFFD40**

7	6	5	4	3	2	1	0
CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0 <sup>1</sup>

- - - - - - - -

CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0 <sup>1</sup>
------	-------	----	------	------	------	------	-------------------

COMMAND CONTROL

PERIPHERAL CHIP SELECT

**NOTES:**

1. The PCS0 bit represents the dual-function PCS0/ $\overline{SS}$ .

Command RAM is used by the QSPI when in master mode. The CPU writes one byte of control information to this segment for each QSPI command to be executed. The QSPI cannot modify information in command RAM.

Command RAM consists of 16 bytes. Each byte is divided into two fields. The peripheral chip-select field enables peripherals for transfer. The command control field provides transfer options.

A maximum of 16 commands can be in the queue. Queue execution by the QSPI proceeds from the address in NEWQP[3:0] through the address in ENDQP[3:0]. (Both of these fields are in SPCR2).

**CONT — Continue**

- 0 = Control of chip selects returned to PORTQS after transfer is complete.
- 1 = Peripheral chip selects remain asserted after transfer is complete.

**BITSE — Bits per Transfer Enable**

- 0 = 8 bits
- 1 = Number of bits set in BITS[3:0] field of SPCR0

**DT — Delay after Transfer**

The QSPI provides a variable delay at the end of serial transfer to facilitate the interface with peripherals that have a latency requirement. The delay between transfers is determined by the SPCR1 DTL[6:0] field.

**DSCK — PCS to SCK Delay**

- 0 = PCS valid to SCK transition is one-half SCK.
- 1 = SPCR1 DSCKL[6:0] field specifies delay from PCS valid to SCK.

**PCS[3:0] — Peripheral Chip Select**

Use peripheral chip-select bits to select an external device for serial data transfer. More than one peripheral chip select can be activated at a time, and more than one peripheral chip can be connected to each PCS pin, provided that proper fanout is observed.

## 5.5.4 Operating Modes

The QSPI operates in either master or slave mode. Master mode is used when the MCU originates data transfers. Slave mode is used when an external device initiates serial transfers to the MCU through the QSPI. Switching between the modes is controlled by MSTR in SPCR0. Before entering either mode, appropriate QSM and QSPI registers must be properly initialized.

In master mode, the QSPI executes a queue of commands defined by control bits in each command RAM queue entry. Chip-select pins are activated, data is transmitted from transmit RAM and received into receive RAM.

In slave mode, operation proceeds in response to  $\overline{SS}$  pin activation by an external bus master. Operation is similar to master mode, but no peripheral chip selects are generated, and the number of bits transferred is controlled in a different manner. When the QSPI is selected, it automatically executes the next queue transfer to exchange data with the external device correctly.

Although the QSPI inherently supports multi-master operation, no special arbitration mechanism is provided. A mode fault flag (MODF) indicates a request for SPI master arbitration. System software must provide arbitration. Note that unlike previous SPI systems, MSTR is not cleared by a mode fault being set, nor are the QSPI pin output drivers disabled. The QSPI and associated output drivers must be disabled by clearing SPE in SPCR1.

## 5.6 SCI Submodule

The SCI submodule is used to communicate with external devices through 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.

### 5.6.1 SCI Pins

There are two unidirectional pins associated with the SCI. The SCI controls the transmit data (TXD) pin when enabled, whereas the receive data (RXD) pin remains a dedicated input pin to the SCI. TXD is available as a general-purpose I/O pin when the SCI transmitter is disabled. When used for I/O, TXD can be configured either as input or output, as determined by QSM register DDRQS.

**Table 42** shows SCI pins and their functions.

**Table 42 SCI Pins**

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

### 5.6.2 SCI Registers

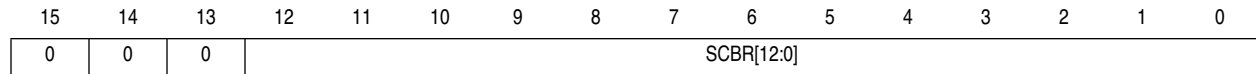
The SCI programming model includes QSM global and pin control registers, and four SCI registers. There are two SCI control registers, one status register, and one data register. All registers can be read or written at any time by the CPU.

Changing the value of SCI control bits during a transfer operation may disrupt operation. Before changing register values, allow the transmitter to complete the current transfer, then disable the receiver and transmitter. Status flags in the SCSR may be cleared at any time.



**SCCR0** — SCI Control Register 0

**\$YFFC08**



RESET:



SCCR0 contains a baud rate selection parameter. Baud rate must be set before the SCI is enabled. The CPU can read and write this register at any time.

Bits [15:13] — Not Implemented

**SCBR[12:0]** — Baud Rate

SCI baud rate is programmed by writing a 13-bit value to BR. The baud rate is derived from the MCU system clock by a modulus counter.

The SCI receiver operates asynchronously. An internal clock is necessary to synchronize with an 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. The SCI determines the position of bit boundaries from transitions within the received waveform, and adjusts sampling points to the proper positions within the bit period. Receiver sampling rate is always 16 times the frequency of the SCI baud rate, which is calculated as follows:

$$\text{SCI Baud Rate} = \frac{\text{System Clock}}{(32)(\text{SCBR}[12:0])}$$

or

$$\text{SCBR}[12:0] = \frac{\text{System Clock}}{32 \times \text{SCI Baud Rate Desired}}$$

where SCBR[12:0] is in the range {1, 2, 3, ..., 8191}

Writing a value of zero to SCBR[12:0] disables the baud rate generator.

**Table 43** lists the SCBR[12:0] settings for standard and maximum baud rates using a 14.44 MHz system clock.

**Table 43 SCI Baud Rates**

Nominal Baud Rate	Actual Rate with 14.44 MHz Clock	SCBR[12:0] Value
64	64.0	\$1B8B
110	110.0	\$1006
300	300.0	\$05E0
600	600.1	\$02F0
1200	1200.1	\$0178
2400	2400.3	\$00BC
4800	4800.5	\$005E
9600	9601.1	\$002F
19200	18802.1	\$0018
38400	37604.2	\$000C
76800	75208.3	\$0006
Maximum rate	451250.0	\$0001

## SCCR1 — SCI Control Register 1

\$YFFC0A

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

SCCR1 contains SCI configuration parameters. The CPU can read and write this register at any time. The SCI can modify RWU in some circumstances. In general, interrupts enabled by these control bits are cleared by reading SCSR, then reading (for receiver status bits) or writing (for transmitter status bits) SCDR.

Bit 15 — Not Implemented

LOOPS — Loop Mode

0 = Normal SCI operation, no looping, feedback path disabled

1 = Test SCI operation, looping, feedback path enabled

LOOPS controls a feedback path on the data serial shifter. When loop mode is enabled, SCI transmitter output is fed back into the receive serial shifter. TXD is asserted (idle line). Both transmitter and receiver must be enabled before entering loop mode.

WOMS — Wired-OR Mode for SCI Pins

0 = If configured as an output, TXD is a normal CMOS output.

1 = If configured as an output, TXD is an open-drain output.

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 TXD is used as a general-purpose input pin, WOMS has no effect.

ILT — Idle-Line Detect Type

0 = Short idle-line detect (start count on first one)

1 = Long idle-line detect (start count on first one after stop bit(s))

PT — Parity Type

0 = Even parity

1 = Odd parity

When parity is enabled, PT determines whether parity is even or odd for both the receiver and the transmitter.

PE — Parity Enable

0 = SCI parity disabled

1 = SCI parity enabled

PE determines whether parity is enabled or disabled for both the receiver and the transmitter. 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 44** lists the available choices.

**Table 44 Parity Enable Data Bit Selection**

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

**M — Mode Select**

- 0 = SCI frame: one start bit, eight data bits, one stop bit (10 bits total)
- 1 = SCI frame: one start bit, nine data bits, one stop bit (11 bits total)

**WAKE — Wakeup by Address Mark**

- 0 = SCI receiver awakened by idle-line detection
- 1 = SCI receiver awakened by address mark (last bit set)

**TIE — Transmit Interrupt Enable**

- 0 = SCI TDRE interrupts inhibited
- 1 = SCI TDRE interrupts enabled

**TCIE — Transmit Complete Interrupt Enable**

- 0 = SCI TC interrupts inhibited
- 1 = SCI TC interrupts enabled

**RIE — Receiver Interrupt Enable**

- 0 = SCI RDRF interrupt inhibited
- 1 = SCI RDRF interrupt enabled

**ILIE — Idle-Line Interrupt Enable**

- 0 = SCI IDLE interrupts inhibited
- 1 = SCI IDLE interrupts enabled

**TE — Transmitter Enable**

- 0 = SCI transmitter disabled (TXD pin may be used as I/O)
- 1 = SCI transmitter enabled (TXD pin dedicated to SCI transmitter)

The transmitter retains control of the TXD pin until completion of any character transfer that was in progress when TE is cleared.

**RE — Receiver Enable**

- 0 = SCI receiver disabled (status bits inhibited)
- 1 = SCI receiver enabled

**RWU — Receiver Wakeup**

- 0 = Normal receiver operation (received data recognized)
- 1 = Wakeup mode enabled (received data ignored until awakened)

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.

**SBK — Send Break**

- 0 = Normal operation
- 1 = Break frame(s) transmitted after completion of current frame

SBK provides the ability to transmit a break code from the SCI. If the SCI is transmitting when SBK is set, it will transmit continuous frames of zeros after it completes the current frame, until SBK is cleared. If SBK is toggled (one to zero in less than one frame interval), the transmitter sends only one or two break frames before reverting to idle line or beginning to send data.

**SCSR — SCI Status Register**

**\$YFFC0C**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
NOT USED								TDRE	TC	RDRF	RAF	IDLE	OR	NF	FE	PF

RESET:

1	1	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---

SCSR contains flags that show SCI operational conditions. These flags can be cleared either by hardware or by a special acknowledgment sequence. The sequence consists of SCSR read with flags set, followed by SCDR read (write in the case of TDRE and TC). A long-word read can consecutively access both SCSR and SCDR. This action clears receive status flag bits that were set at the time of the read, but does not clear TDRE or TC flags.

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 cleared. SCSR must be read again with the bit set. Also, SCDR must be written or read before the status bit is cleared.

Reading either byte of SCSR causes all 16 bits to be accessed. Any status bit already set in either byte will be cleared on a subsequent read or write of register SCDR.

#### TDRE — Transmit Data Register Empty Flag

0 = Register TDR still contains data to be sent to the transmit serial shifter.

1 = A new character can now be written to the transmit data register.

TDRE is set when the byte in the transmit data register is transferred to the transmit serial shifter. If TDRE is zero, transfer has not occurred and a write to the transmit data register will overwrite the previous value. New data is not transmitted if the transmit data register is written without first clearing TDRE.

#### TC — Transmit Complete Flag

0 = SCI transmitter is busy

1 = SCI transmitter is idle

TC is set when the transmitter finishes shifting out all data, queued preambles (mark/idle line), or queued breaks (logic zero). The interrupt can be cleared by reading SCSR when TC is set and then by writing the transmit data register of SCDR.

#### RDRF — Receive Data Register Full Flag

0 = Receive data register is empty or contains previously read data.

1 = Receive data register contains new data.

RDRF is set when the content of the receive serial shifter is transferred to the receive data register. If one or more errors are detected in the received word, flag(s) NF, FE, and/or PF are set within the same clock cycle.

#### RAF — Receiver Active Flag

0 = SCI receiver is idle

1 = SCI receiver is busy

RAF indicates whether the SCI receiver is busy. It is set when the 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.

#### IDLE — Idle-Line Detected Flag

0 = SCI receiver did not detect an idle-line condition.

1 = SCI receiver detected an idle-line condition.

IDLE is disabled when RWU in SCCR1 is set. IDLE is set when the SCI receiver detects the idle-line condition specified by ILT in SCCR1. If cleared, IDLE will not set again until after RDRF is set. RDRF is set when a break is received, so that a subsequent idle line can be detected.

#### OR — Overrun Error Flag

0 = RDRF is cleared before new data arrives.

1 = RDRF is not cleared before new data arrives.

OR is set when a new byte is ready to be transferred from the receive serial shifter to the receive data register, and RDRF is still set. Data transfer is inhibited until OR is cleared. Previous data in receive data register remains valid, but data received during overrun condition (including the byte that set OR) is lost.

### NF — Noise Error Flag

0 = No noise detected on the received data

1 = Noise occurred on the received data

NF is set when the SCI receiver detects noise on a valid start bit, on any data bit, or on a stop bit. It is not set by noise on the idle line or on invalid start bits. Each bit is sampled three times. If none of the three samples are the same logic level, the majority value is used for the received data value, and NF is set. NF is not set until an entire frame is received and RDRF is set.

### FE — Framing Error Flag

0 = No framing error on the received data.

1 = Framing error or break occurred on the received data.

FE is set when the SCI receiver detects a zero where a stop bit was to have occurred. FE is not set until the entire frame is received and RDRF is set. A break can also cause FE to be set. It is possible to miss a framing error if RXD happens to be at logic level one at the time the stop bit is expected.

### PF — Parity Error Flag

0 = No parity error on the received data

1 = Parity error occurred on the received data

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.

### SCDR — SCI Data Register

**\$YFFC0E**

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

SCDR contains two data registers at the same address. The receive data register is a read-only register that contains data received by the SCI. The data comes into the receive serial shifter and is transferred to the receive data register. The transmit data register is a write-only register that contains data to be transmitted. The data is first written to the transmit data register, then transferred to the transmit serial shifter, where additional format bits are added before transmission. R[7:0]/T[7:0] contain either the first eight data bits received when SCDR is read, or the first eight data bits to be transmitted when SCDR is written. R8/T8 are used when the SCI is configured for 9-bit operation. When it is configured for 8-bit operation, they have no meaning or effect.

## 6 Configurable Timer Module 6

The configurable timer module 6 (CTM6) belongs to a family of timer modules for the Motorola Modular Microcontroller Family. The timer architecture is modular relative to the number of time bases (counter submodules) and channels (action submodules or timer I/O pins) that are included.

The CTM6 consists of several submodules which are located on either side of the CTM6 internal submodule bus (SMB). All data and control signals within the CTM6 are passed over this bus. The SMB is connected to the outside world via the bus interface unit submodule (BIUSM), which is connected to the intermodule bus (IMB), and subsequently the CPU. This configuration allows the CPU to access the data and control registers in each CTM6 submodule on the SMB. Four local time base buses TBB[1:4], each 16-bits wide, are used to transfer timing information from counters to action submodules. Each CTM6 submodule can be connected to two TBBs. **Figure 17** shows a block diagram of the CTM6.

### 6.1 Overview

The time base bus system connects the four counter submodules to the eleven double-action submodules (DASMs) and eight single-action submodules (SASM) channels.

The bus interface unit submodule (BIUSM) allows all the CTM6 submodules to communicate to the IMB via the submodule bus (SMB).

The counter prescaler submodule (CPSM) generates six different clock frequencies which can be used by any counter submodule. This submodule is contained within the BIUSM.

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. One FCSM is contained in the CTM6.

The modulus counter submodule (MCSM) is an enhancement of the FCSM. A modulus register gives the additional flexibility of recycling the counter at a count other than 64K clock cycles. Three MCSMs are contained in the CTM6.

The single action submodule (SASM) provides an input capture and an output compare for each of two bidirectional pins. A total of four SASMs (eight channels) are contained in the CTM6.

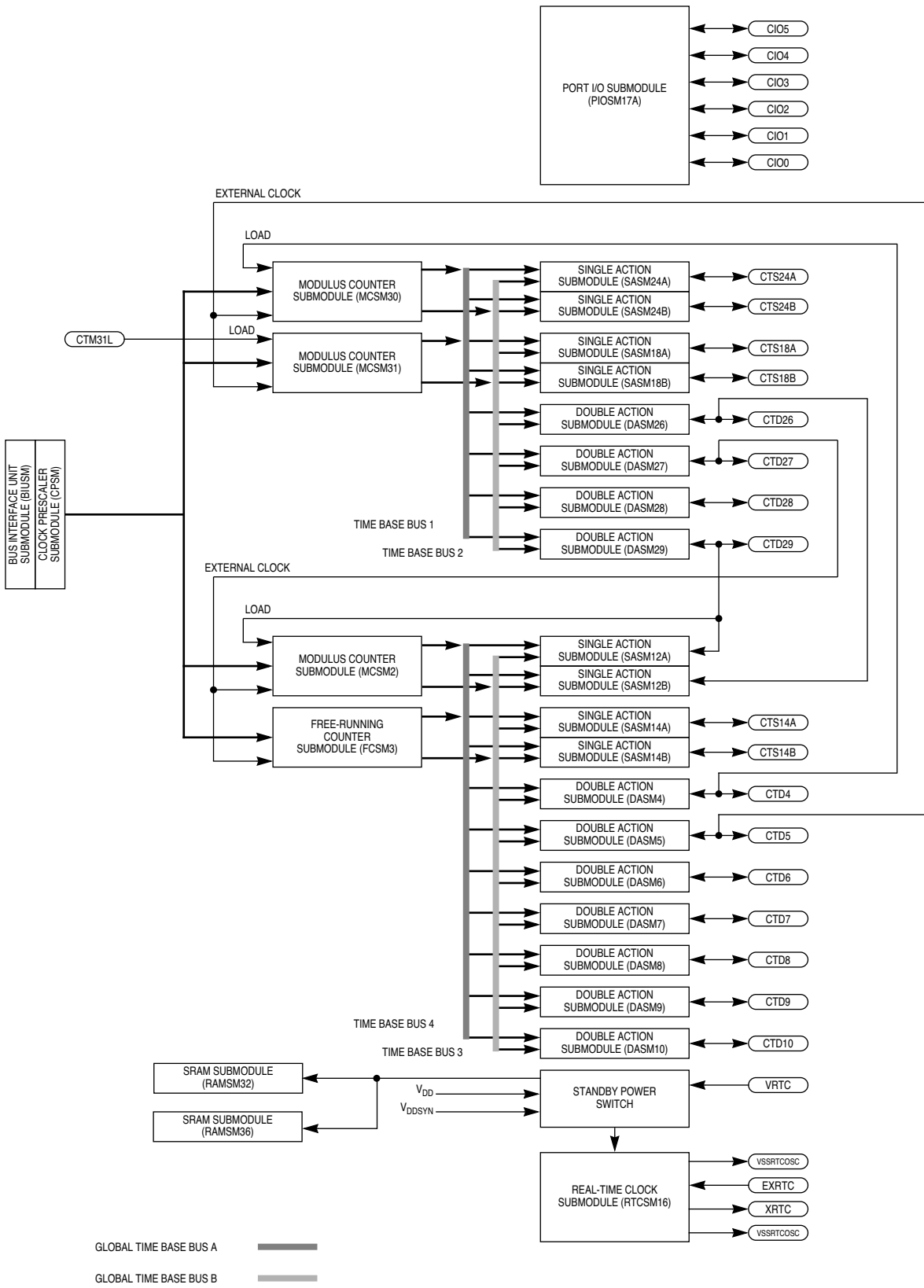
The double-action submodule (DASM) provides two 16-bit input captures or two 16-bit output compare functions that can occur automatically without software intervention. Eleven DASMs are contained in the CTM6.

The real-time clock submodule (RTCSM) provides a real time clock function independent of other CTM6 submodules. This time counter is driven by a dedicated low frequency oscillator (32.768 kHz) for low power consumption.

A parallel port I/O submodule (PIOSM) handles up to eight input/output pins. Each pin of the PIOSM may be programmed as an input or an output under software control. One PIOSM is contained in the CTM6.

The static RAM submodule (RAMSM) provides 32 bytes (16 words) of contiguous memory locations and takes the address space of four standard submodules. The RAMSM works as a stand-by memory. Two RAMSMs are contained in the CTM6.

The standby power switch changes the power source of the RTCSM prescaler, counters, and oscillator, as well as the two RAMSMs to  $V_{RTC}$  when  $V_{DD}$  drops below its minimum specified value.



CTM6 BLOCK

Figure 17 CTM6 Block Diagram

## 6.2 Address Map

The CTM6 address map occupies 512 bytes. All CTM6 registers are addressable in supervisor space only. **Table 45** shows the CTM6 register.

**Table 45 CTM6 Address Map**

Address	15	0
\$YFF400 <sup>1</sup>	BIUSM Module Configuration Register (BIUMCR)	
\$YFF402	BIUSM Test Register (BIUTEST)	
\$YFF404	BIUSM Time Base Register (BIUTBR)	
\$YFF406	Reserved	
\$YFF408	CPSM Control Register (CPCR)	
\$YFF40A	CPSM Test Register (CPTR)	
\$YFF40C – \$YFF40E	Reserved	
\$YFF410	MCSM2 Status/Interrupt/Control Register (MCSM2SICR)	
\$YFF412	MCSM2 Counter Register (MCSM2CNT)	
\$YFF414	MCSM2 Modulus Latch (MCSM2ML)	
\$YFF416	Reserved	
\$YFF418	FCSM3 Status/Interrupt/Control Register (FCSM3SIC)	
\$YFF41A	FCSM3 Counter Register (FCSM3CNT)	
\$YFF41C – \$YFF41E	Reserved	
\$YFF420	DASM4 Status/Interrupt/Control Register (DASM4SIC)	
\$YFF422	DASM4 Register A (DASM4A)	
\$YFF424	DASM4 Register B (DASM4B)	
\$YFF426	Reserved	
\$YFF428	DASM5 Status/Interrupt/Control Register (DASM5SIC)	
\$YFF42A	DASM5 Register A (DASM5A)	
\$YFF42C	DASM5 Register B (DASM5B)	
\$YFF42E	Reserved	
\$YFF430	DASM6 Status/Interrupt/Control Register (DASM6SIC)	
\$YFF432	DASM6 Register A (DASM6A)	
\$YFF434	DASM6 Register B (DASM6B)	
\$YFF436	Reserved	
\$YFF438	DASM7 Status/Interrupt/Control Register (DASM7SIC)	
\$YFF43A	DASM7 Register A (DASM7A)	
\$YFF43C	DASM7 Register B (DASM7B)	
\$YFF43E	Reserved	
\$YFF440	DASM8 Status/Interrupt/Control Register (DASM8SIC)	
\$YFF442	DASM8 Register A (DASM8A)	
\$YFF444	DASM8 Register B (DASM8B)	
\$YFF446	Reserved	
\$YFF448	DASM9 Status/Interrupt/Control Register (DASM9SIC)	
\$YFF44A	DASM9 Register A (DASM9A)	
\$YFF44C	DASM9 Register B (DASM9B)	
\$YFF44E	Reserved	



**Table 45 CTM6 Address Map (Continued)**

<b>Address</b>	<b>15</b>	<b>0</b>
\$YFF450	DASM10 Status/Interrupt/Control Register (DASM10SIC)	
\$YFF452	DASM10 Register A (DASM10A)	
\$YFF454	DASM10 Register B (DASM10B)	
\$YFF456 – \$YFF45E	Reserved	
\$YFF460	SASM12 Status/Interrupt/Control Register A (SIC12A)	
\$YFF462	SASM12 Data Register A (S12DATA)	
\$YFF464	SASM12 Status/Interrupt/Control Register B (SIC12B)	
\$YFF466	SASM12 Data Register B (S12DATB)	
\$YFF468 – \$YFF46E	Reserved	
\$YFF470	SASM14 Status/Interrupt/Control Register A (SIC14A)	
\$YFF472	SASM14 Data Register A (S14DATA)	
\$YFF474	SASM14 Status/Interrupt/Control Register B (SIC14B)	
\$YFF476	SASM14 Data Register B (S14DATB)	
\$YFF478 – \$YFF47E	Reserved	
\$YFF480	RTCSM16 Status/Interrupt/Control Register (RTC16SIC)	
\$YFF482	RTCSM16 Prescaler Register (R16PRR)	
\$YFF484	RTCSM16 32-Bit Free-Running Counter High (R16FRCH)	
\$YFF486	RTCSM16 32-Bit Free-Running Counter Low (R16FRCL)	
\$YFF488	PIOSM17A I/O Port Register (PIO17A)	
\$YFF48A – \$YFF48E	Reserved	
\$YFF490	SASM18 Status/Interrupt/Control Register A (SIC18A)	
\$YFF492	SASM18 Data Register A (S18DATA)	
\$YFF494	SASM18 Status/Interrupt/Control Register B (SIC18B)	
\$YFF496	SASM18 Data Register B (S18DATB)	
\$YFF498 – \$YFF4BE	Reserved	
\$YFF4C0	SASM24 Status/Interrupt/Control Register A (SIC24A)	
\$YFF4C2	SASM24 Data Register A (S24DATA)	
\$YFF4C4	SASM24 Status/Interrupt/Control Register B (SIC24B)	
\$YFF4C6	SASM24 Data Register B (S24DATB)	
\$YFF4C8 – \$YFF4CE	Reserved	
\$YFF4D0	DASM26 Status/Interrupt/Control Register (DASM26SIC)	
\$YFF4D2	DASM26 Register A (DASM26A)	
\$YFF4D4	DASM26 Register B (DASM26B)	
\$YFF4D6	Reserved	
\$YFF4D8	DASM27 Status/Interrupt/Control Register (DASM27SIC)	
\$YFF4DA	DASM27 Register A (DASM27A)	
\$YFF4DC	DASM27 Register B (DASM27B)	
\$YFF4DE	Reserved	
\$YFF4E0	DASM28 Status/Interrupt/Control Register (DASM28SIC)	
\$YFF4E2	DASM28 Register A (DASM28A)	
\$YFF4E4	DASM28 Register B (DASM28B)	
\$YFF4E6	Reserved	

**Table 45 CTM6 Address Map (Continued)**

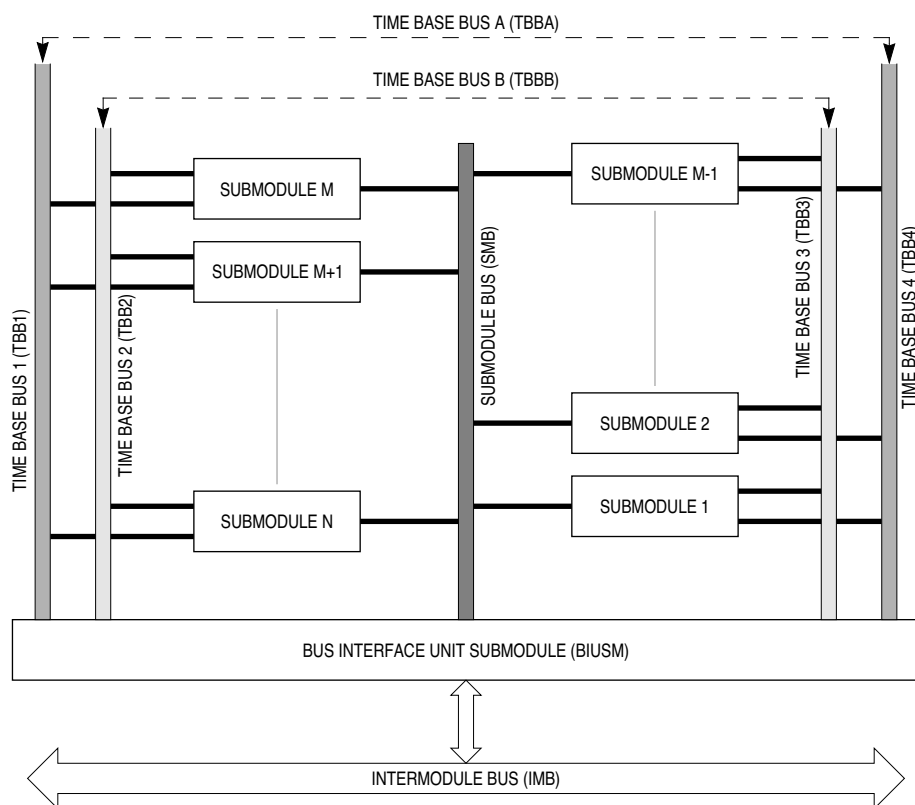
<b>Address</b>	<b>15</b>	<b>0</b>
\$YFF4E8	DASM29 Status/Interrupt/Control Register (DASM29SIC)	
\$YFF4EA	DASM29 Register A (DASM29A)	
\$YFF4EC	DASM29 Register B (DASM29B)	
\$YFF4EE	Reserved	
\$YFF4F0	MCSM30 Status/Interrupt/Control Register (MCSM30SIC)	
\$YFF4F2	MCSM30 Counter Register (MCSM30CNT)	
\$YFF4F4	MCSM30 Modulus Latch (MCSM30ML)	
\$YFF4F6	Reserved	
\$YFF4F8	MCSM31 Status/Interrupt/Control Register (MCSM31SIC)	
\$YFF4FA	MCSM31 Counter Register (MCSM31CNT)	
\$YFF4FC	MCSM31 Modulus Latch (MCSM31ML)	
\$YFF4FE	Reserved	
\$YFF500 – \$YFF51E	RAMSM32 RAM	
\$YFF520 – \$YFF53E	RAMSM36 RAM	
\$YFF540 – \$YFF5FE	Reserved	

NOTES:

1. Y = M111, where M is the logic state of the module mapping (MM) bit in the SIMLCR.

### 6.3 Time Base Bus System

The time base bus system is composed of four 16-bit buses: TBB1, TBB2, TBB3 and TBB4. They are arranged so that each CTM6 submodule can be connected to two time base buses. **Figure 18** shows that CTM6 submodules numbered 1 to M-1 can be connected to TBB3 and TBB4. CTM6 submodules M to N can be connected to TBB1 and TBB2. Control bits within each CTM6 submodule allow the software to connect the submodule to the desired time base bus(es).



CTM TBB BLOCK

**Figure 18 Time Base Bus Configuration**

As shown in **Figure 18**, each CTM submodule can access one of two global time base buses. TBB1 and TBB4 are collectively referred to as global time base bus TBBA. Likewise, TBB2 and TBB3 are collectively referred to as global time base bus TBBB. **Table 46** shows which time base buses are available for each CTM6 submodule.

**Table 46 CTM6 Time Base Bus Allocation**

Submodule	Global/Local Time Base Bus Allocation		Submodule	Global/Local Time Base Bus Allocation	
	Global Bus A	Global Bus B		Global Bus A	Global Bus B
MCSM2	TBB4	TBB3	SASM18	TBB1	TBB2
FCSM3	TBB4	TBB3	SASM24	TBB1	TBB2
DASM4 – 10	TBB4	TBB3	DASM26 – 29	TBB1	TBB2
SASM12 – 14	TBB4	TBB3	MCSM30 – 31	TBB1	TBB2

Time base buses are used to transfer timing information from counters to action submodules. Each CTM6 submodule can either be a clock source module (and drive one or two of the time base buses) or an action submodule (and read and react to the timing information on the time base buses).

The time base buses are precharge/discharge type buses with wired-OR capability, so that no hardware damage occurs when several counters are driving the same bus at the same time.

## 6.4 Bus Interface Unit Submodule (BIUSM)

The BIUSM connects the SMB to the IMB and allows the CTM6 submodules to communicate with the CPU. The BIUSM also communicates interrupt requests from the CTM6 submodules to the IMB, and transfers the interrupt level, arbitration bit and vector number to the CPU during the interrupt acknowledge cycle.

### 6.4.1 BIUSM Registers

The BIUSM contains a module configuration register, a time base register, and a test register. The BIUSM register block always occupies the first four register locations in the CTM6 register space and cannot be relocated within the CTM6 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.

#### BIUMCR — BIU Module Configuration Register

**\$YFF400**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP	FRZ	0	VECT[7:6]	IARB[2:0]			0	0	TBR51	0	0	0	0	0	TBR50

RESET:

0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0

#### STOP — Stop Enable

The STOP bit, while asserted, completely stops operation of the CTM6. The BIUSM continues to operate to allow the CPU access to submodule registers. The CTM6 remains stopped until reset or until the STOP bit is negated by the CPU.

0 = Allows operation of the CTM6

1 = Stops operation of the CTM6

#### FRZ — FREEZE Assertion Response

0 = Ignore IMB FREEZE signal

1 = CTM6 stops when IMB FREEZE signal is asserted

#### NOTE

Some submodules may validate this signal with internal enable bits.

#### Bit 13 — Not Implemented

#### VECT[7:6] — Interrupt Vector Base Number Field

This bit field selects the interrupt vector base number for the CTM6. Of the eight 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 VECT[7:6]. This places the CTM6 vectors in one of four possible positions in the interrupt vector table. Refer to **Table 47**.

**Table 47 Interrupt Vector Base Number Bit Field**

VECT7	VECT6	Resulting Vector Base Number
0	0	\$00
0	1	\$40
1	0	\$80
1	1	\$C0

### IARB[2:0] — Interrupt Arbitration Field

This bit field and the IARB3 bit within each submodule provide 15 different interrupt arbitration 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, preventing the module from arbitrating during an IACK cycle. If no arbitration takes place during the IACK cycle, the SIML generates a spurious interrupt, indicating to the system that the interrupt arbitration number has not been initialized.

The CTM6 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 CTM6 interrupt requests. Therefore, CTM6 submodule interrupts can be prioritized with requests from other modules at the same interrupt level. IARB[2:0] are cleared by reset.

### Bits[7:6], [4:1] — Not Implemented

### TBRS1, TBRS0 — Time Base Register Bus Select Bits

These bits specify which time base bus is accessed when the time base register (BIUTBR) is read. Refer to **Table 48**.

**Table 48 Time Base Register Bus Select Bits**

TBRS1	TBRS0	Time Base Bus
0	0	TBB1
0	1	TBB2
1	0	TBB3
1	1	TBB4

### BIUTEST — BIUSM Test Register

**\$YFF402**

BIUTEST is used during factory testing of the CTM6. Accesses to BIUTEST must be made while the MCU is in test mode.

### BIUTBR — BIUSM Time Base Register

**\$YFF404**

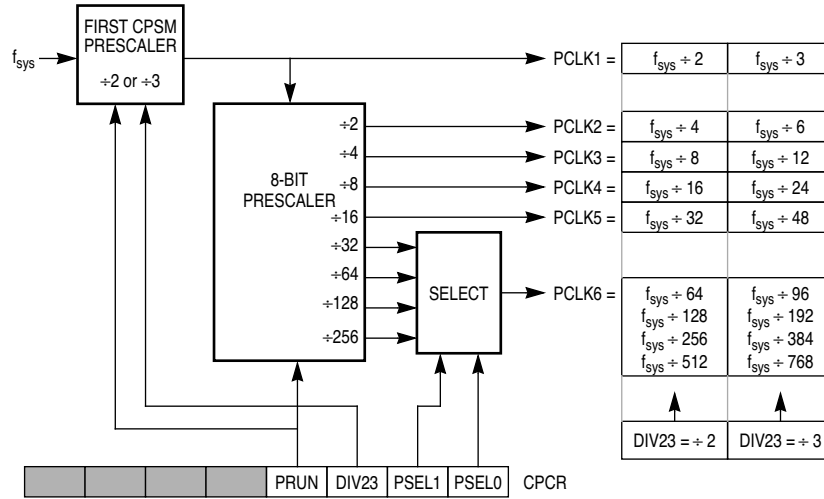
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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 BIUMCR. Writing to BIUTBR has no effect during normal operation.

## 6.5 Counter Prescaler Submodule (CPSM)

The counter prescaler submodule (CPSM) is a programmable divider system that provides the CTM6 counters with a choice of six clock signals (PCLKx) derived from the sixth frequency MCU system clock ( $f_{sys}$ ). Five of these frequencies are derived from a fixed divider chain. The divide ratio is software selectable from a choice of four divide ratios.

The CPSM is contained within the BIUSM. **Figure 19** shows a block diagram of the CPSM.



CTM CPSM BLOCK

**Figure 19 CPSM Block Diagram**

### 6.5.1 CPSM Registers

The CPSM contains a control register and a test 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.

#### CPCR — CPSM Control Register

**\$YFF408**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED												PRUN	DIV23	PSEL[1:0]	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

#### 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 CTM6 submodules to be synchronized. It is cleared by reset.

- 0 = Prescaler is held in reset and is not running
- 1 = Prescaler is running

#### DIV23 — Divide By 2/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 by reset.

- 0 = First prescaler stage divides by two
- 1 = First prescaler stage divides by three

#### PSEL[1:0] — Prescaler Division Ratio Select Field

This bit field selects the division ratio of the programmable prescaler output signal (PCLK6). Refer to **Table 49**.

**Table 49 Prescaler Division Ratio Select Field**

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

**CPTR — CPSM Test Register**

**\$YFF40A**

CPTR is used during factory testing of the CTM6. Accesses to CPTR must be made while the MCU is in test mode.

**6.6 Clock Sources for Counter Submodules**

One of seven clock sources can be chosen for each counter submodule. Five of them are fixed prescaler taps derived from the system clock:  $\div 2$ ,  $\div 4$ ,  $\div 8$ ,  $\div 16$ , and  $\div 32$ . A sixth prescaler tap is software selectable as the system clock divided by 64, 128, 256, or 512. An alternate prescaler option provides fixed prescaler taps of the system clock divided by 3, 6, 12, 24, and 48. In this case, the software selectable tap is the system clock divided by 96, 192, 384, or 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 a clock frequency to be selected that is not based on the MCU system clock. Alternately, the external clock input allows a counter submodule to be used for pulse or event counting.

**NOTE**

The external clock inputs for MCSM30 and MCSM31 are tied to the I/O pin CTD5 for DASM5. The external clock inputs for MCSM2 and FCSM3 are tied to the I/O pin CTD27 for DASM27.

**6.7 Free-Running Counter Submodule (FCSM)**

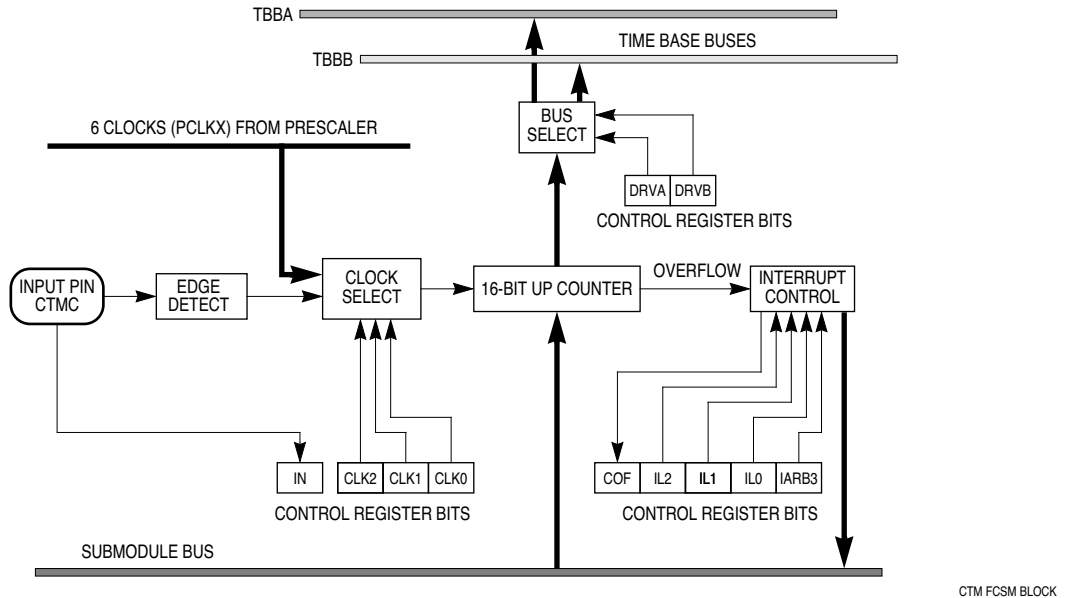
The free-running counter submodule (FCSM) has a 16-bit up counter with an associated clock source selector, selectable time-base bus drivers, software writable control registers, software readable status bits, and interrupt logic. When the 16-bit up counter overflows from \$FFFF to \$0000, an optional overflow interrupt may be generated.

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.

One FCSM is contained in the CTM6. **Figure 20** shows a block diagram of the FCSM.

## NOTE

In order to count, the FCSM requires the CPSM clock signals to be present. On coming out of reset, the FCSM does 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 CTM6 submodules to be synchronized.



**Figure 20 FCSM Block Diagram**

### 6.7.1 FCSM Registers

The FCSM contains a status/interrupt/control register and a counter register. All unused bits and reserved address locations return zero when read. Writing to unused bits and reserved address locations has no effect.

#### FCSM3SIC — FCSM Status/Interrupt/Control Register

**\$YFF418**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COF	IL[2:0]			IARB3	0	DRVA	DRVB	IN	0	0	0	0	CLK[2:0]		

RESET:

0   0   0   0   0   0   0   0   U   0   0   0   0   0   0   0

#### COF — Counter Overflow Flag

This status flag indicates whether or not a counter overflow has occurred. An overflow is defined to be the transition of the counter from \$FFFF to \$0000. If the IL field is non-zero, an interrupt request is generated when the COF bit is set.

0 = Counter overflow has not occurred

1 = Counter overflow has occurred

This flag is set only by hardware and cleared only by software or a system reset. To clear the flag, first read the register with COF set to one, then write a zero to the bit. COF is cleared only if no overflow occurs between the read and write operations.



### IL[2:0] — Interrupt Level

Setting IL[2:0] to a non-zero value causes the FCSM to request an interrupt of the selected level when the COF bit sets. If IL[2:0] = %000, no interrupt will be requested when COF sets. These bits can be read or written at any time and are cleared by reset.

### IARB3 — Interrupt Arbitration Bit 3

This bit works in conjunction with IARB[2:0] in the BIUMCR. Each module that generates interrupt requests on the IMB must have a unique value in the arbitration field. 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. Refer to **6.4.1 BIUSM Registers** for more information on IARB[2:0].

### DRV[A:B] — Drive Time Base Bus

This bit field contains read/write bits that control the connection of the FCSM to the time base buses A and B. These bits are cleared by reset. Refer to **Table 52**.

**Table 50 Drive Time Base Bus Field**

DRVA	DRVB	Bus Selected
0	0	No time base bus driven
0	1	Time base bus B is driven
1	0	Time base bus A is driven
1	1	Both time base buses A and B are driven

#### WARNING

Two time base buses should not be driven at the same time.

### IN — Input Pin Status Bit

This read-only status bit reflects the logic state of the FCSM input pin. Writing to this bit has no effect, nor does reset.

#### NOTE

The clock input of FCSM3 is internally connected to I/O pin CTD27 of DASM27 and will read the state of that pin.

### CLK[2:0] — Counter Clock Select

These read/write control bits select one of six internal clock signals (PCLK[1:6]) or one of two external conditions on the external clock input pin. Maximum frequency of the external clock signals is  $f_{\text{sys}}/4$ . Refer to **Table 54**.

**Table 51 Counter Clock Select Field**

CLK2	CLK1	CLK0	Free-Running Counter Clock Source
0	0	0	PCLK1 ( $f_{\text{sys}} \div 2$ or $f_{\text{sys}} \div 3$ )
0	0	1	PCLK2 ( $f_{\text{sys}} \div 4$ or $f_{\text{sys}} \div 6$ )
0	1	0	PCLK3 ( $f_{\text{sys}} \div 8$ or $f_{\text{sys}} \div 12$ )
0	1	1	PCLK4 ( $f_{\text{sys}} \div 16$ or $f_{\text{sys}} \div 24$ )
1	0	0	PCLK5 ( $f_{\text{sys}} \div 32$ or $f_{\text{sys}} \div 48$ )
1	0	1	PCLK6 ( $f_{\text{sys}} \div 64$ or $f_{\text{sys}} \div 768$ )
1	1	0	External clock input, falling edge
1	1	1	External clock input, rising edge

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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. A read returns the current value of the counter. A write loads the counter with the specified value. The counter then begins incrementing from this new value.

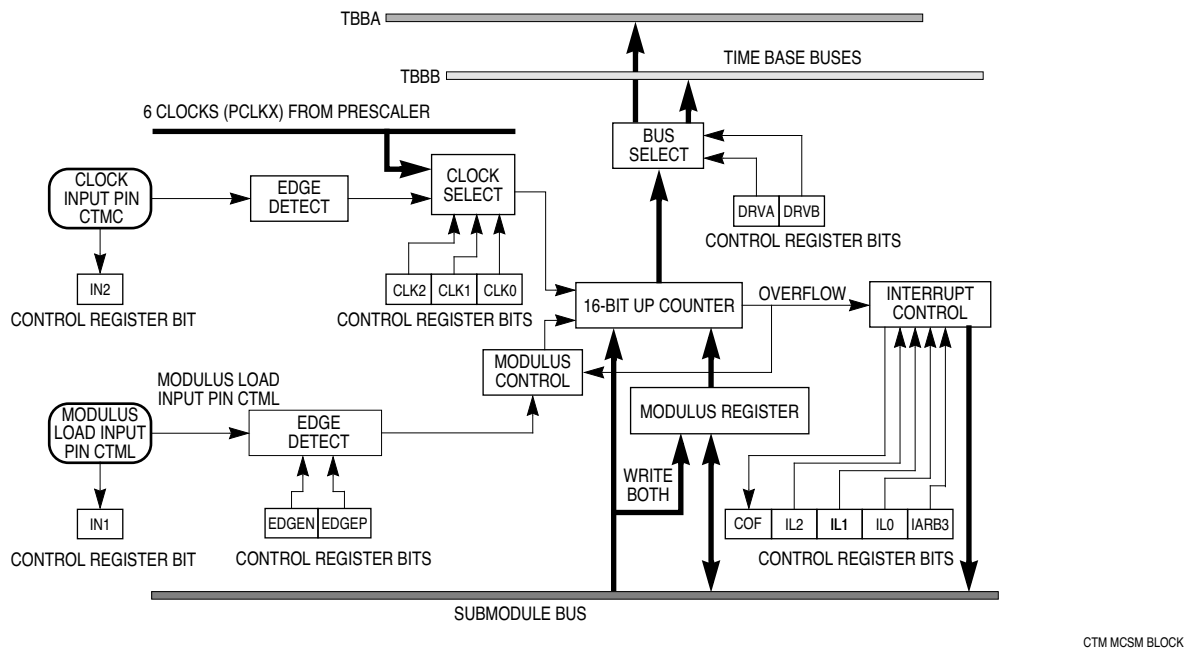
**6.8 Modulus Counter Submodule (MCSM)**

The modulus counter submodule (MCSM) is an enhancement of the FCSM. The MCSM contains 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. A modulus latch gives the additional flexibility of recycling the counter at a count other than 64-Kbyte clock cycles. The state of the modulus latch is transferred to the counter when an overflow occurs or when a user-specified edge transition occurs on an external input pin. In addition, a write to the modulus counter simultaneously loads both the counter and the modulus latch with the specified value. The counter then begins incrementing from this new value.

Three MCSMs are contained in the CTM6. **Figure 21** shows a block diagram of the MCSM.

**NOTE**

In order to count, the MCSM requires the CPSM clock signals to be present. On coming out of reset, the MCSM does 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 CTM6 submodules to be synchronized.



**Figure 21 MCSM Block Diagram**

## 6.8.1 MCSM Registers

The MCSM contains a status/interrupt/control register, a counter, and a modulus latch. All unused bits and reserved address locations return zero when read. Writing to unused bits and reserved address locations has no effect. The CTM6 contains three MCSMs, each with its own set of registers.

**MCSM2SIC** — MCSM2 Status/Interrupt/Control Register **\$YFF410**  
**MCSM30SIC** — MCSM30 Status/Interrupt/Control Register **\$YFF4F0**  
**MCSM31SIC** — MCSM31 Status/Interrupt/Control Register **\$YFF4F8**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COF	IL[2:0]			IARB3	0	DRVA	DRVB	IN2	IN1	EDGEN	EDGEPE	0	CLK[2:0]		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

### COF — Counter Overflow Flag

This status flag 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 \$FFFF to \$xxxx, where \$xxxx is the value contained in the modulus latch. If the IL[2:0] field is non-zero, an interrupt request is generated when the COF bit is set.

- 0 = Counter overflow has not occurred
- 1 = Counter overflow has occurred

This flag is set only by hardware and cleared only by software or by a system reset. To clear the flag, the software must first read the register with COF set to one, then write a zero to the bit. COF is cleared only if no overflow occurs between the read and write operations.

### IL[2:0] — Interrupt Level

Setting IL[2:0] to a non-zero value causes the MCSM to request an interrupt of the selected level when the COF bit sets. If IL[2:0] = %000, no interrupt will be requested when COF sets. These bits can be read or written at any time and are cleared by reset.

### IARB3 — Interrupt Arbitration Bit 3

This bit works in conjunction with IARB[2:0] in the BIUMCR. Each module that generates interrupt requests on the IMB must have a unique value in the arbitration field. 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. Refer to **6.4.1 BIUSM Registers** for more information on IARB[2:0].

### DRV[A:B] — Drive Time Base Bus

This bit field contains read/write bits that control the connection of the MCSM to time base buses A and B. These bits are cleared by reset. Refer to **Table 52**.

**Table 52 Drive Time Base Bus Field**

DRVA	DRVB	Bus Selected
0	0	No time base bus is driven
0	1	Time base bus B is driven
1	0	Time base bus A is driven
1	1	Both time base buses A and B are driven

### WARNING

Two time base buses should not be driven at the same time.

## IN2 — Clock Input Pin Status

This read-only status bit reflects the logic state of the clock input pin. Writing to this bit has no effect, nor does reset.

### NOTE

The clock input of MCSM2 is internally connected to I/O pin CTD27 for DASM27 and will read the state of that pin. The clock inputs of MCSM30 and MCSM31 are internally connected to I/O pin CTD5 for DASM5 and will read the state of that pin.

## IN1 — Modulus Load Input Pin Status

This read-only status bit reflects the logic state of the modulus load input pin. Writing to this bit has no effect, nor does reset.

### NOTE

The load input of MCSM2 is internally connected to I/O pin CTD29 for DASM29 and will read the state of that pin. The load input of MCSM30 is internally connected to I/O pin CTD4 for DASM4 and will read the state of that pin. The load input of MCSM31 is available externally on the CTM31L pin.

## EDGEN, EDGEF — Modulus Load Edge Sensitivity Bits

These read/write control bits select the sensitivity of the edge detection circuitry on the modulus load pin CTML. Refer to **Table 53**.

**Table 53 Modulus Load Edge Sensitivity**

EDGEN	EDGEF	IN1 Edge Detector Sensitivity
0	0	None
0	1	Positive edge only
1	0	Negative edge only
1	1	Positive and negative edge

## CLK[2:0] — Counter Clock Select

These read/write control bits select one of six internal clock signals (PCLK[1:6]) or one of two external conditions on the external clock input pin (rising edge or falling edge). The maximum frequency of the external clock signals is  $f_{sys}/4$ . Refer to **Table 54**.

**Table 54 Counter Clock Select**

CLK2	CLK1	CLK0	Free-Running Counter Clock Source
0	0	0	PCLK1 ( $f_{sys} \div 2$ or $f_{sys} \div 3$ )
0	0	1	PCLK2 ( $f_{sys} \div 4$ or $f_{sys} \div 6$ )
0	1	0	PCLK3 ( $f_{sys} \div 8$ or $f_{sys} \div 12$ )
0	1	1	PCLK4 ( $f_{sys} \div 16$ or $f_{sys} \div 24$ )
1	0	0	PCLK5 ( $f_{sys} \div 32$ or $f_{sys} \div 48$ )
1	0	1	PCLK6 ( $f_{sys} \div 64$ or $f_{sys} \div 768$ )
1	1	0	External clock input, falling edge
1	1	1	External clock input, rising edge

**NOTE**

The clock input of MCSM2 is internally connected to I/O pin CTD27 for DASM27 and will read the state of that pin. The clock inputs of MCSM30 and MCSM31 are internally connected to I/O pin CTD5 for DASM5 and will read the state of that pin.

**MCSM2CNT** — MCSM2 Counter Register **\$YFF412**  
**MCSM30CNT** — MCSM30 Counter Register **\$YFF4F2**  
**MCSM31CNT** — MCSM31 Counter Register **\$YFF4FA**

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

The MCSM counter register is a read/write register. A read returns the current value of the counter. A write simultaneously loads both the counter and the MCSM modulus latch with the specified value. The counter then begins incrementing from this new value.

**MCSM2ML** — MCSM2 Modulus Latch **\$YFF414**  
**MCSM30ML** — MCSM30 Modulus Latch **\$YFF4F4**  
**MCSM31ML** — MCSM31 Modulus Latch **\$YFF4FC**

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

The MCSM modulus latch register is a read/write register. A read returns the current value of the latch. A write pre-loads the latch with a new value that the modulus counter will begin counting from when the next load condition occurs.

**6.9 Single Action Submodule (SASM)**

The single action submodule (SASM) provides two identical channels, each having its own input/output pin, but sharing the same interrupt logic, priority level, and arbitration number. Each channel can be configured independently to perform either input capture or output compare. **Table 55** shows the different operational modes.

**Table 55 SASM Operational Modes**

Mode	Function
IC <sup>1</sup>	Input capture either on a rising or falling edge or as a read-only input port
OC <sup>2</sup>	Output compare
OCT <sup>2</sup>	Output compare and toggle
OP <sup>2</sup>	Output port

NOTES:

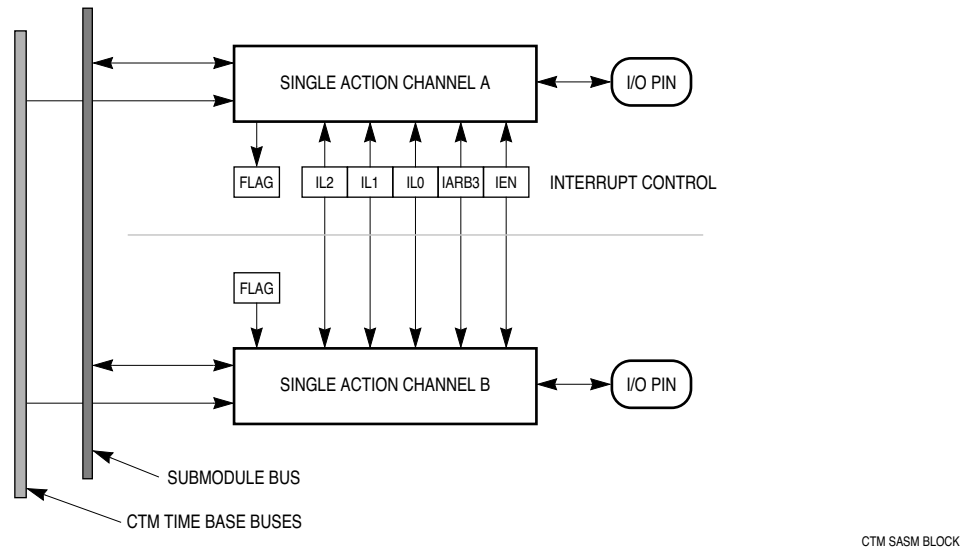
1. When a channel is operating in IC mode, the IN bit in the SIC register reflects the logic state of the corresponding input pin (after being Schmitt triggered and synchronized).
2. When a channel is operating in OC, OCT, or OP mode, the IN bit in the SIC register reflects the logic state of the output of the output flip-flop.

**NOTE**

All of the functions associated with one pin are called a SASM channel.

The SASM can perform a single timing action (input capture or output compare) before software intervention is required. Each channel includes a 16-bit 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 can be set to either toggle when an output compare occurs or to transfer a software-provided bit value to the output pin. In either input capture or output compare mode, each channel can be programmed to generate an interrupt. One of the two incoming time-base buses may be selected for each channel. Each channel can also work as a simple I/O pin.

A total of four SASMs (eight channels) are contained in the CTM6. **Figure 22** shows a block diagram of the SASM.



**Figure 22 SASM Block Diagram**

**6.9.1 SASM Registers**

The SASM contains two status/interrupt/control registers (A and B) and two data registers (A and B). All unused bits and reserved address locations return zero when read. Writing to unused bits and reserved address locations has no effect. The CTM6 contains four SASMs, each with its own set of registers.

**SIC12A, SIC14A** — SASM Status/Interrupt/Control Register A **\$YFF460, \$YFF470**  
**SIC18A, SIC24A** — SASM Status/Interrupt/Control Register A **\$YFF490, \$YFF4C0**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLAG	IL[2:0]			IARB3	IEN	0	BSL	IN	0	FORCE	EDOUT	0	0	MODE[1:0]	
RESET:															
0	0	0	0	0	0	0	0	U	0	0	0	0	0	0	0

**SIC12B, SIC14B** — SASM Status/Interrupt/Control Register B  
**SIC18B, SIC24B** — SASM Status/Interrupt/Control Register B

**\$YFF464, \$YFF474**  
**\$YFF494, \$YFF4C4**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLAG	0	0	0	0	IEN	0	BSL	IN	0	FORCE	EDOUT	0	0	MODE[1:0]	
RESET:															
0	0	0	0	0	0	0	0	U	0	0	0	0	0	0	0

SICA and SICB contain the control and status bits for SASM channels A and B, respectively. SICA also contains the IL[2:0] interrupt level field and IARB3 interrupt arbitration bit 3 for both SASM channels A and B.

**FLAG** — Event Flag

FLAG indicates whether or not an input capture or output compare event has occurred. If the IL[2:0] field is non-zero, and IEN is set, an interrupt request is generated when FLAG is set.

- 0 = An input capture or output compare event has not occurred
- 1 = An input capture or output compare event has occurred

**Table 56** shows the event flag status during different modes.

**Table 56 Event Flag Status Conditions**

Mode	Status Description
IC	If a subsequent input capture event occurs while FLAG is set, the new value is latched and FLAG remains set.
OC	If a subsequent output compare event occurs while FLAG is set, the compare occurs normally and FLAG remains set.
OCT	If a subsequent output compare event occurs while FLAG is set, the output signal toggles normally and FLAG remains set.
OP	If a subsequent internal compare event occurs while FLAG is set, the compare occurs normally and FLAG remains set.

FLAG is set only by hardware and cleared only by software or by a system reset. To clear this bit, first read the register with FLAG set to one, then write a zero to the bit.

**NOTE**

The flag clearing mechanism works 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 cleared.

**IL[2:0]** — Interrupt Level

Setting IP[2:0] to a non-zero value causes the SASM to request an interrupt when the FLAG bit sets. If IL[2:0] = %000, no interrupts will be requested when FLAG sets.

**NOTE**

This field affects both SASM channels, not just channel A.

**IARB3** — Interrupt Arbitration Bit 3

This bit works in conjunction with IARB[2:0] in the BIUMCR. Each module that generates interrupt requests on the IMB must have a unique value in the arbitration field. 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. Refer to **6.4.1 BIUSM Registers** for more information on IARB[2:0].

#### NOTE

This bit field affects both SASM channels, not just channel A.

#### IEN — Interrupt Enable

This control bit enables interrupts when FLAG is set and the IL[2:0] field is non-zero.

0 = Interrupts disabled

1 = Interrupts enabled

#### BSL — Time Base Bus Select

This control bit selects the time base bus connected to the SASM.

0 = Time base bus A selected

1 = Time base bus B selected

#### 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 flip-flop.

The IN bit is a read-only bit. Reset has no effect on this bit.

#### NOTE

The input of SASM12A is internally connected to I/O pin CTD29 and will read the state of that pin. The input of SASM12B is internally connected to I/O pin CTD26 and will read the state of that pin.

#### FORCE — Force Compare Control

In the IC and OP modes, FORCE is not used and writing to it has no effect.

In the OC and OCT modes, FORCE is used by software to cause the output flip-flop (and the output pin) to behave as though an output compare had occurred. In OC and OCT mode, setting FORCE causes the value of EDOUT to be transferred to the output of the output flip-flop. 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.

0 = No action

1 = Force output flip-flop to behave as if an output compare has occurred

FORCE is cleared by reset and always reads as zero.

#### NOTE

FLAG is not affected by the use of the FORCE bit.

#### EDOUT — Edge Detect and Output Level

In IC mode, EDOUT is used to select the edge that triggers the input capture circuitry.

0 = Input capture on falling edge

1 = Input capture on rising edge

In OC and OCT 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 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 OP mode, the value of EDOUT is output to the corresponding pin. Reading EDOUT returns the previous value written.

#### MODE[1:0] — SASM Operating Mode

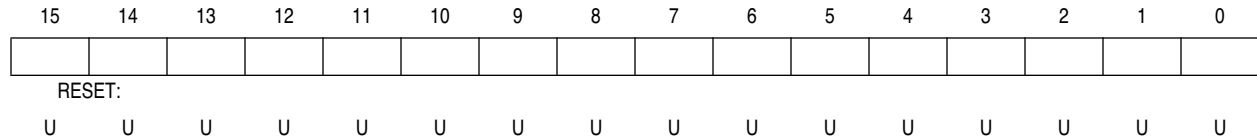
This bit field selects the mode of operation for the SASM channel. Refer to **Table 57**.



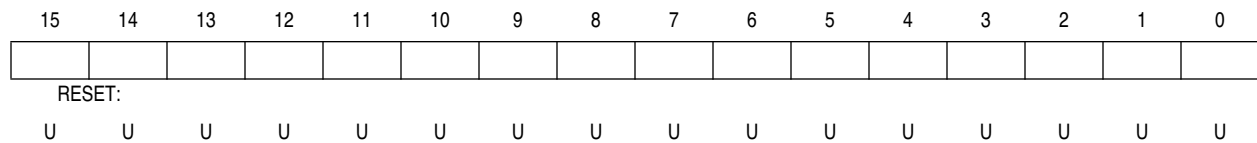
**Table 57 SASM Operating Mode Select**

MODE1	MODE2	SASM Channel Operating Mode
0	0	Input capture (IC)
0	1	Output port (OP)
1	0	Output compare (OC)
1	1	Output compare and toggle (OCT)

**S12DATA, S14DATA** — SASM Data Register A **\$YFF462, \$YFF472**  
**S18DATA, S24DATA** — SASM Data Register A **\$YFF492, \$YFF4C2**



**S12DATB, S14DATB** — SASM Data Register B **\$YFF464, \$YFF474**  
**S18DATB, S24DATB** — SASM Data Register B **\$YFF494, \$YFF4C4**



SDATA and SDATB are the data registers associated with SASM channels A and B, respectively. In IC mode, SDATA and SDATB contain the last captured value. In the OC, OCT and OP modes, SDATA and SDATB are loaded with the value of the next output compare.

### 6.10 Double-Action Submodule (DASM)

The double-action submodule (DASM) provides two 16-bit input capture 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 user-specified edges. The output flip flop is set by one of the output compare signals and reset by the other one. The DASM input capture and the output compare modes may optionally generate interrupts. Software determines which of the two incoming time-base buses is used for input captures or output compares.

Six operating modes allow software to use DASM input capture and output compare functions to perform pulse-width measurement, period measurement, single pulse generation, and continuous pulse width modulation, as well as standard input capture and output compare. The DASM can also work as a single I/O pin. DASM operation is determined by the mode select bit field MODE[3:0] in the DASM status/interrupt/control (DASMSIC) register. **Table 58** shows the different DASM modes of operation.

**Table 58 DASM Modes of Operation**

Mode	Description of Mode
DIS	Disabled — I/O pin is placed in a high impedance state
IPWM	Input pulse width measurement — Capture on leading and the trailing edges of an input pulse
IPM	Input period measurement — Capture on two consecutive rising or falling edges of an input pulse
IC	Input capture — Capture on user-specified edge
OCB	Output compare, flag set on channel B match — Generate leading and trailing edges of an output pulse and set flag on second edge
OCAB	Output compare, flag set on channels A and B match — Generate leading and trailing edges of an output pulse and set flag on both edges
OPWM	Output pulse width modulation — Generate continuous PWM output with 7, 9, 11, 12, 13, 14, 15, or 16 bits of resolution

The DASM is composed of two timing channels (A and B), an output flip-flop, an input edge detector, some control logic and an interrupt interface. All control and status bits are contained in the DASMSIC register.

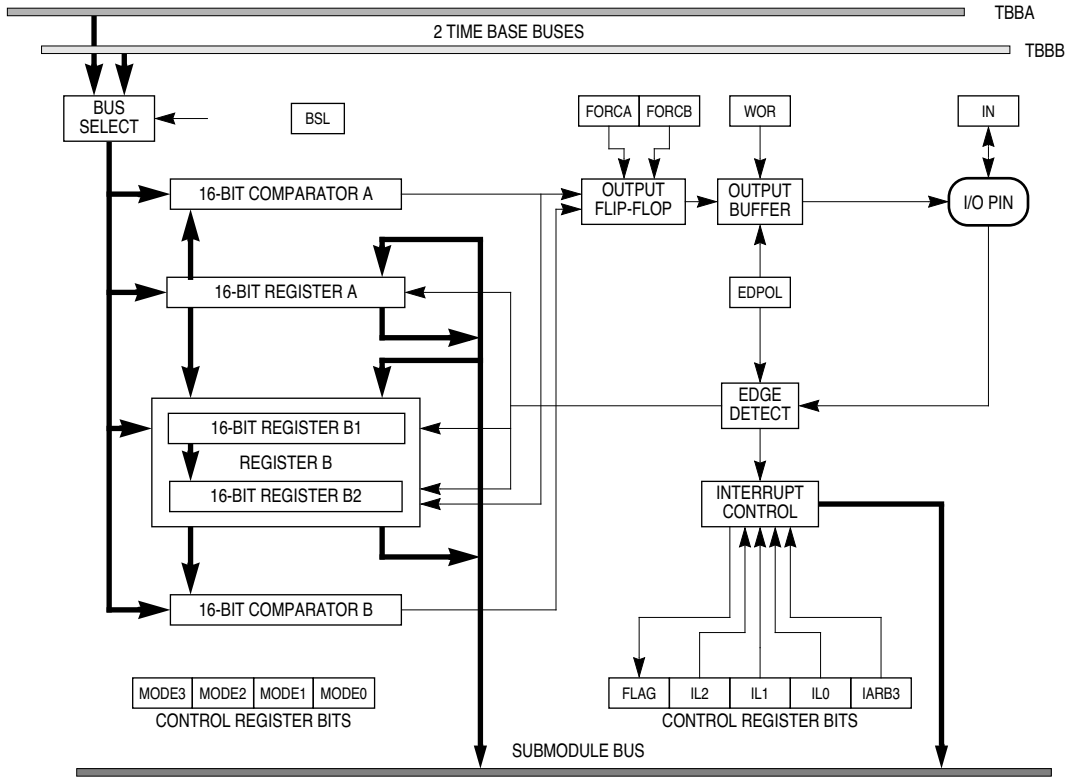
Channel A consists of one 16-bit data register and one 16-bit comparator. To the user, channel B also appears to consist of one 16-bit data register and one 16-bit comparator, though internally, channel B has two data registers (B1 and B2). The operating mode determines which register is accessed by the software. Refer to **Table 59**.

**Table 59 Channel B Data Register Access**

Mode	Data Register
IPWM, IPM, 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.
OCA, OCAB	Registers A and B2 are used to define the output pulse. Register B1 is not used in these modes.
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 CTM6 submodule bus.

Eleven DASMs are contained in the CTM6. **Figure 23** shows a block diagram of the DASM.



CTM DASM BLOCK

Figure 23 DASM Block Diagram

6.10.1 DASM Registers

The DASM contains one status/interrupt/control register and two data registers (A and B). All unused bits and reserved address locations return zero when read. Writing to unused bits and reserved address locations has no effect. The CTM6 contains 11 DASMs, each with its own set of registers.

- DASM4SIC, DASM5SIC** — DASM Status/Interrupt/Control Register **\$YFF420, \$YFF428**
- DASM6SIC, DASM7SIC** — DASM Status/Interrupt/Control Register **\$YFF430, \$YFF438**
- DASM8SIC, DASM9SIC** — DASM Status/Interrupt/Control Register **\$YFF440, \$YFF448**
- DASM10SIC, DASM26SIC** — DASM Status/Interrupt/Control Register **\$YFF450, \$YFF4D0**
- DASM27SIC, DASM28SIC** — DASM Status/Interrupt/Control Register **\$YFF4D8, \$YFF4E0**
- DASM29SIC** — DASM Status/Interrupt/Control Register **\$YFF4E8**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLAG	IL[2:0]			IARB3	0	WOR	BSL	IN	FORCA	FORCB	EDPOL	MODE[3:0]			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

## FLAG — Event Flag

This status bit indicates whether or not an input capture or output compare event has occurred. If the IL[2:0] field is non-zero, an interrupt request is generated when FLAG is set.

0 = An input capture or output compare event has not occurred

1 = An input capture or output compare event has occurred

**Table 56** shows the event flag status during different modes.

**Table 60 Event Flag Status Conditions**

Mode	Status Description
DIS	FLAG bit is cleared
IPWM	FLAG bit is set each time there is a capture on channel A
IPM	FLAG bit is set each time there is a capture on channel A, except for the first time
IC	FLAG bit is set each time there is a capture on channel A
OCB	FLAG bit is set each time there is a successful comparison on channel B
OCAB	FLAG bit is set each time there is a successful comparison on either channel A or B
OPWM	FLAG bit is set each time there is a successful comparison on channel A

FLAG is set only by hardware and cleared by software or by a system reset. To clear the bit, first read the register with FLAG set to one, then write a zero to the bit. Placing the DASM in DIS mode will also clear the flag.

### NOTE

The flag clearing mechanism works 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.

## IL[2:0] — Interrupt Level

Setting IL[2:0] to a non-zero value causes the DASM to request an interrupt when the FLAG bit sets. If IL[2:0] = %000, no interrupt will be requested when FLAG sets.

## IARB3 — Interrupt Arbitration Bit 3

This bit works in conjunction with IARB[2:0] in the BIUMCR. Each module that generates interrupt requests on the IMB must have a unique value in the arbitration field. 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. Refer to **6.4.1 BIUSM Registers** for more information on IARB[2:0].

## WOR — Wired-OR Mode

In the DIS, IPWM, IPM and IC modes, WOR is not used. Reading this bit returns the value that was previously written.

In the OCB, OCAB and OPWM modes, WOR selects whether the output buffer is configured for normal or open drain operation.

0 = Output buffer operates in normal mode

1 = Output buffer operates in open drain mode

## BSL — Bus Select

This bit selects the time base bus connected to the DASM.

0 = DASM is connected to time base bus A.

1 = DASM is connected to time base bus B.

#### 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.

#### FORCA — Force A

In the OCB, OCAB and OPWM modes, FORCA allows software to force the output flip-flop to behave as if a successful comparison had occurred on channel A (except that the FLAG bit is not set). Writing a one to FORCA sets the output flip-flop; writing a zero has no effect.

In the DIS, IPWM, IPM and IC modes, FORCA is not used and writing to it has no effect.

FORCA is cleared by reset, and always reads as zero.

#### NOTE

Writing a one to both FORCA and FORCB simultaneously resets the output flip-flop.

#### FORCB — Force B

In the OCB, OCAB and OPWM modes, FORCB allows software to force the output flip-flop to behave as if a successful comparison had occurred on channel B (except that the FLAG bit is not set). Writing a one to FORCB sets the output flip-flop, writing a zero has no effect.

In the DIS, IPWM, IPM and IC modes, FORCB is not used and writing to it has no effect.

FORCB is cleared by reset, and always reads as zero.

#### NOTE

Writing a one to both FORCA and FORCB simultaneously resets the output flip-flop.

#### EDPOL — Edge Polarity

EDPOL selects different options depending on the DASM operating mode. Refer to **Table 61**.

**Table 61 Edge Polarity**

MODE	EDPOL	Function
DIS	X	EDPOL is not used in DIS mode
IPWM	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
IPM, IC	0	Channel A captures on a rising edge
	1	Channel A captures on a falling edge
OCB, OCAB, OPWM	0	A compare on channel A sets the output pin to logic one A compare on channel B clears the output pin to logic zero
	1	A compare on channel A clears the output pin to logic zero A compare on channel B sets the output pin to logic one

#### MODE[3:0] — DASM Mode Select

This bit field selects the operating mode of the DASM. Refer to **Table 62**.

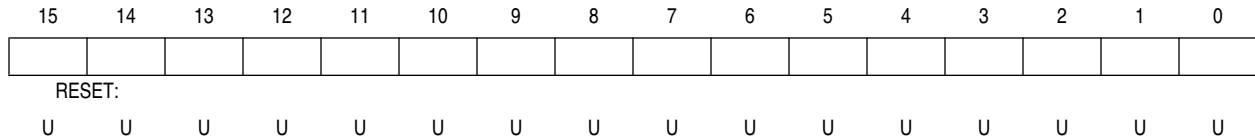
**NOTE**

To avoid spurious interrupts, DASM interrupts should be disabled before changing the operating mode.

**Table 62 DASM Mode Select Field**

MODE[3:0]	Bits of Resolution	Time Base Bits Ignored	DASM Operating Mode
0000	—	—	DIS — Disabled
0001	16	—	IPWM — Input pulse width measurement
0010	16	—	IPM — Input measurement period
0011	16	—	IC — Input capture
0100	16	—	OCB — Output compare, flag on B compare
0101	16	—	OCAB — Output compare, flag on A and B compare
011X	—	—	Not used
1000	16	—	OPWM — Output pulse-width modulation
1001	15	15	OPWM — Output pulse-width modulation
1010	14	[15:14]	OPWM — Output pulse-width modulation
1011	13	[15:13]	OPWM — Output pulse-width modulation
1100	12	[15:12]	OPWM — Output pulse-width modulation
1101	11	[15:11]	OPWM — Output pulse-width modulation
1110	9	[15:9]	OPWM — Output pulse-width modulation
1111	7	[15:7]	OPWM — Output pulse-width modulation

**DASM4A, DASM5A** — DASM Data Register A **\$YFF422, \$YFF42A**  
**DASM6A, DASM7A** — DASM Data Register A **\$YFF432, \$YFF43A**  
**DASM8A, DASM9A** — DASM Data Register A **\$YFF442, \$YFF44A**  
**DASM10A, DASM26A** — DASM Data Register A **\$YFF452, \$YFF4D2**  
**DASM27A, DASM28A** — DASM Data Register A **\$YFF4DA, \$YFF4E2**  
**DASM29A** — DASM Data Register A **\$YFF4EA**

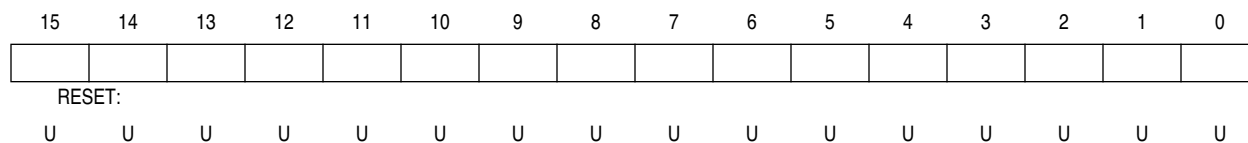


DASMA is the data register associated with channel A. **Table 63** shows how the DASMA is used with the different operating modes.

**Table 63 DASMA Operations**

Mode	DASMA Operation
DIS	DASMA can be accessed to prepare a value for a subsequent mode selection
IPWM	DASMA contains the captured value corresponding to the trailing edge of the measured pulse
IPM	DASMA contains the captured value corresponding to the most recently detected user-specified rising or falling edge
IC	DASMA contains the captured value corresponding to the most recently detected user-specified rising or falling edge
OCB	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.
OCAB	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.
OPWM	DASMA is loaded with the value corresponding to the leading edge of the PWM pulse to be generated.

**DASM4B, DASM5B** — DASM Data Register B \$YFF424, \$YFF42C  
**DASM6B, DASM7B** — DASM Data Register B \$YFF434, \$YFF43C  
**DASM8B, DASM9B** — DASM Data Register B \$YFF444, \$YFF44C  
**DASM10B, DASM26B** — DASM Data Register B \$YFF454, \$YFF4D4  
**DASM27B, DASM28B** — DASM Data Register B \$YFF4DC, \$YFF4E4  
**DASM29B** — DASM Data Register B \$YFF4EC



DASMB is the data register associated with channel B. **Table 64** shows how DASMB is used with the different operating modes. Depending on the mode selected, software access is to register B1 or register B2.

**Table 64 DASMB Operations**

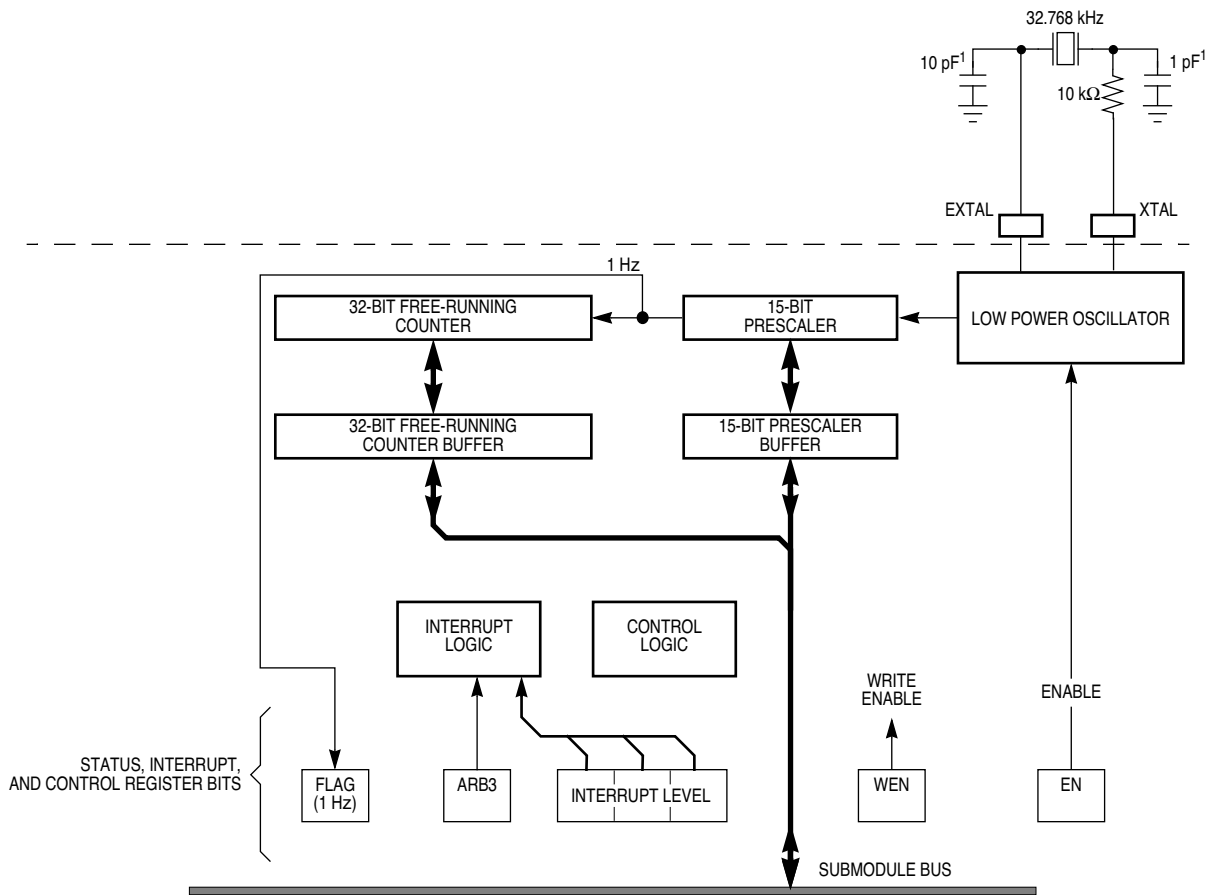
Mode	DASMB Operation
DIS	DASMB can be accessed to prepare a value for a subsequent mode selection. In this mode, register B1 is accessed in order to prepare a value for the OPWM mode. Unused register B2 is hidden and cannot be read, but is written with the same value when register B1 is written.
IPWM	DASMB contains the captured value corresponding to the trailing edge of the measured pulse. In this mode, register B2 is accessed. Buffer register B1 is hidden and cannot be accessed.
IPM	DASMB contains the captured value corresponding to the most recently detected user-specified rising or falling edge. In this mode, register B2 is accessed. Buffer register B1 is hidden and cannot be accessed.
IC	DASMB contains the captured value corresponding to the most recently detected user-specified rising or falling edge. In this mode, register B2 is accessed. Buffer register B1 is hidden and cannot be accessed.
OCB	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.
OCAB	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.
OPWM	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.

### 6.11 Real-Time Clock Submodule (RTCSM) with Low-Power Oscillator

The real-time clock submodule provides a real-time clock independent of other CTM6 submodules. This time counter is driven by a dedicated low frequency oscillator (32.768 kHz) for low power consumption. The RTCSM contains a 15-bit prescaler and a 32-bit free-running counter, from which seconds, minutes, hours and days can be determined. The RTCSM can also generate interrupts at one second intervals. The low-power oscillator, prescaler, and counter portions of the RTCSM may be sustained by a separate power supply ( $V_{RTC}$ ) for battery backup when  $V_{DD}$  is off. The RTCSM and the low-power oscillator can be disabled for minimum power consumption on  $V_{RTC}$  when  $V_{DD}$  is powered off. This is useful for maximizing the shelf life of the standby battery. Refer to **6.14 RTCSM and RAMSM Standby Operation** for more information.

One RTCSM is contained in the CTM6. **Figure 24** shows a block diagram of the RTCSM.





NOTES:

- RESISTANCE AND CAPACITANCE BASED ON A TEST CIRCUIT CONSTRUCTED WITH A DAISHINKU DMX-38 32.768 kHz CRYSTAL. SPECIFIC COMPONENTS MUST BE BASED ON CRYSTAL TYPE. CONTACT CRYSTAL VENDOR FOR EXACT CIRCUIT.

CTM RTC BLOCK

Figure 24 RTCSM Block Diagram

### 6.11.1 RTCSM Registers

The RTCSM contains one status/interrupt/control register, one 15-bit prescaler register and two 32-bit free-running counter registers (high and low). All unused bits and reserved address locations return zero when read. Writing to unused bits and reserved address locations has no effect.

#### RTC16SIC — RTCSM Status/Interrupt/Control Register

\$YFF480

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TICKF	IL[2:0]			IARB3	0	WEN	EN	NOT USED							

RESET:

U 0 0 0 0 0 0 U 0 0 0 0 0 0 0 0

#### TICKF — 1 Hz Clock Tick Flag

TICKF is set each time the 32-bit free-running counter is incremented. Software can clear TICKF by reading the bit as a one, then writing a zero to it. If the IL[2:0] field is non-zero, an interrupt request is generated when TICKF is set.

TICKF is not affected by reset.

**NOTE**

TICKF is only cleared if the 32-bit free-running counter does not increment between reading RTC16SIC with TICKF set to one and then writing TICKF to zero.

**IL[2:0] — Interrupt Level Field**

Setting IL[2:0] to a non-zero value causes the RTCSM to request an interrupt of the selected level when the TICKF bit sets. If IL[2:0] = %000, no interrupt will be requested when TICKF sets.

**IARB3 — Interrupt Arbitration Bit 3**

This bit works in conjunction with IARB[2:0] in the BIUMCR. Each module that generates interrupt requests on the IMB must have a unique value in the arbitration field. 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. Refer to **6.4.1 BIUSM Registers** for more information on IARB[2:0].

**WEN — Write Enable Control**

This bit allows the 15-bit prescaler and the 32-bit free-running counter to be updated. Normally, these are read-only registers. Regular write operations have no effect. When the WEN bit is written to one, it sets a latch that allows the 15-bit prescaler and the 32-bit free-running counter to be written. The latch is automatically reset when the prescaler is written.

To write a new value to the complete counter chain:

- Write a one to the WEN bit.
- Execute a long-word write to the 32-bit free-running counter high (R16FRCH) register.
- Execute a word write to the 15-bit prescaler (R16PRR) register.

WEN cannot be written to one again until the writes to update the prescaler and free-running counter have been completed. The WEN bit always reads as zero.

**EN — RTCSM Enable**

This bit selects whether the RTCSM is running or not.

- 0 = RTCSM is not running
- 1 = RTCSM is running

The EN bit is not affected by reset. If the RTCSM is not to be used, it is recommended that EN be cleared as soon as the MCU comes out of reset.

**R16PRR — RTCSM Prescaler Register**

**\$YFF482**

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

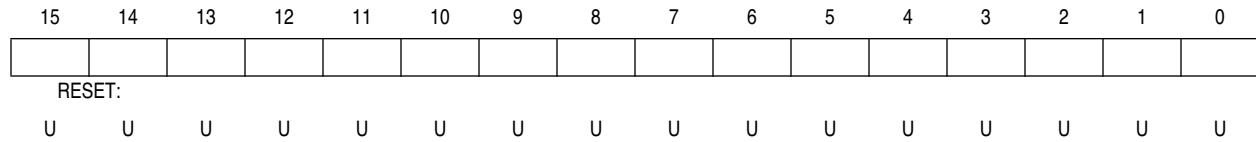
R16PRR contains the synchronized value of the 15-bit prescaler or the value to be loaded into the 15-bit prescaler.

**NOTE**

When the RTCSM is disabled, writing to the 15-bit prescaler and 32-bit free-running counter may give unpredictable results.

**R16FRCH — RTCSM Free-Running Counter High Register**

**\$YFF484**



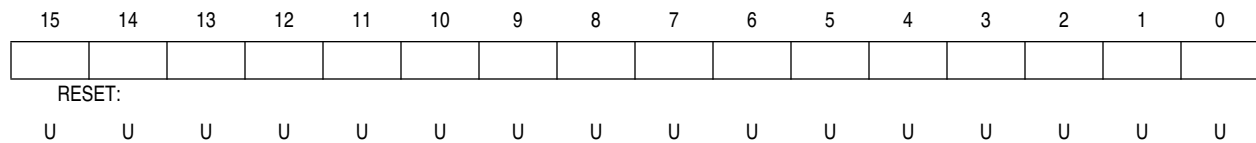
R16FRCH contains the synchronized high word value of the 32-bit free-running counter or the value to be loaded into the high word 32-bit free-running counter.

**NOTE**

When the RTCSM is disabled, writing to the 15-bit prescaler and 32-bit free-running counter may give unpredictable results.

**R16FRCL — RTCSM Free-Running Counter Low Register**

**\$YFF486**



R16FRCL contains the synchronized low word value of the 32-bit free-running counter or the value to be loaded into the low word 32-bit free-running counter.

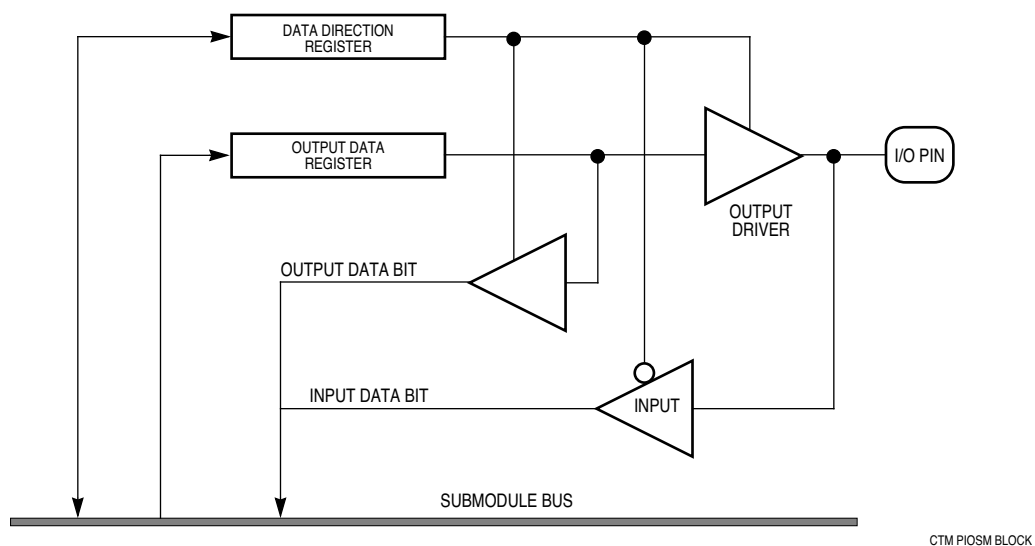
**NOTE**

When the RTCSM is disabled, writing to the 15-bit prescaler and 32-bit free-running counter may give unpredictable results.

**6.12 Parallel Port I/O Submodule (PIOSM)**

The port I/O submodule (PIOSM) provides I/O capability independent of other CTM6 modules. The PIOSM handles up to eight input/output pins.

One PIOSM is contained in the CTM6. **Figure 25** shows a block diagram of the PIOSM.



**Figure 25 PIOSM Block Diagram**

### 6.12.1 PIOSM Register

The PIOSM control register is composed of two 8-bit registers. The upper eight bits contain the data register and the lower eight bits contain the data direction register. Each PIOSM pin may be programmed as an input or an output under software control. The data direction register controls whether the corresponding pins are inputs or outputs.

The PIOSM data register can be read or written by the processor. For pins programmed as outputs, a read of the data register actually reads the value of the output data latch and not the I/O pin.

#### PIO17A — PIOSM Control Register

**\$YFF488**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0	DDR7	DDR6	DDR5	DDR4	DDR3	DDR2	DDR1	DDR0
RESET:															
0	0	U	U	U	U	U	U	0	0	0	0	0	0	0	0

CTIO[7:6] are not bonded to pins on the MC68338. When one of these signals is configured as an input, a read of the corresponding data bit always returns a zero. When one of these signals is configured as an output, a read of the corresponding data bit returns the value stored in the output data latch.

#### NOTE

Care should be taken when a single word write cycle is used to modify the data register and data direction register of the PIOSM. Undesired glitches can occur on pins that change from inputs to outputs and vice versa. To avoid this, first use a byte write cycle to modify the data register then use another byte write cycle to modify the data direction register.

### 6.13 Static RAM Submodule (RAMSM)

The static RAM submodule (RAMSM) provides 32 bytes (16 words) of contiguous memory locations and is not relocatable. It is especially useful for storage of variables and system parameters that must be maintained when the rest of the MCU is powered down. Data can be read or written in bytes, words, or long words. RAMSM locations are not affected by reset.

The CTM6 has two RAMSMs. **Table 65** shows the RAMSM address locations.

**Table 65 RAMSM Address Locations**

Static RAM Submodule	Address
32	\$YFF500–51E
36	\$YFF520–53E

### 6.14 RTCSM and RAMSM Standby Operation

The standby power switch in CTM6 monitors  $V_{DD}$  and selects either  $V_{DD}$  and  $V_{DDSYN}$  or  $V_{RTC}$  for the power source of the RTCSM and RAMSMs, depending on the level of  $V_{DD}$ .

When  $V_{DD}$  is within the specified operating range, the RTCSM low-power oscillator is powered by  $V_{DDSYN}$  and the RAMSMs are powered by  $V_{DD}$ .  $V_{DD}$  also provides power to the digital logic portion of the RTCSM, therefore both  $V_{DD}$  and  $V_{DDSYN}$  must be kept equal to each other for normal operation.

When  $V_{DD}$  and  $V_{DDSYN}$  are powered down, the submodules are powered by  $V_{RTC}$  and are in standby mode. In standby mode, the RTCSM continues to keep time if enabled. However, updates to the 15-bit prescaler and 32-bit free-running counter buffer registers are halted in order to conserve power. All

RTCSM registers are write protected in standby mode to prevent loss of data in runaway situations. For the same reason, the RAMSMs are also write protected in standby mode.

If the standby mode function is not required in a given application,  $V_{RTC}$  should be powered from the  $V_{DD}$  and  $V_{DDSYN}$  supply. Unpredictable operation of the RAMSMs and RTCSM may result otherwise.

### 6.15 CTM6 Interrupts

The CTM6 is able to request numerous interrupts on the IMB. Each submodule that is able to request interrupts can do so with any of seven levels. Each submodule that is able to request interrupts includes a 3-bit level number and a 1-bit arbitration number that is initialized by software.

The 3-bit level number selects which of seven interrupt signals on the IMB are driven by that submodule to create an interrupt request. Of the four priority bits provided on the IMB during arbitration among the modules, one of them comes from the chosen submodule, and the BIUSM provides the other three. Thus, the CTM6 responds to two of the possible 15 arbitration numbers.

During the IMB arbitration process, the BIUSM manages the separate arbitration among the CTM6 submodules to determine which submodule should 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.

When the IARB number is not unique for a given module, simultaneous interrupts are prioritized in hardware according to the vector number or the submodule interrupt arbitration sequence number shown in **Table 66**. Following the interrupt arbitration process, the CTM6 provides an 8-bit vector number. Six of the eight bits are provided by the submodules. A submodule can identify two separate interrupt sources with unique interrupt vectors. The two high-order bits of the 8-bit vector are provided by the BIUSM. The six low-order vector bits identify the highest priority interrupt request pending in the CTM6 at the beginning of the arbitration cycle.

**Table 66 CTM6 Interrupt Priority and Vector/Pin Allocation**

Submodule Name	Submodule Base Address <sup>1</sup>	Submodule Interrupt Vector Number <sup>2</sup>	Submodule Interrupt Arbitration Sequence Number <sup>3</sup>
BIUSM	\$YFF400	None	None
CPSM	\$YFF408	None	None
MCSM2	\$YFF410	xx000010	2
FCSM3	\$YFF418	xx000011	3
DASM4	\$YFF420	xx000100	4
DASM5	\$YFF428	xx000101	5
DASM6	\$YFF430	xx000110	6
DASM7	\$YFF438	xx000111	7
DASM8	\$YFF440	xx001000	8
DASM9	\$YFF448	xx001001	9
DASM10	\$YFF450	xx001010	10
SASM12	\$YFF460	xx001100	12
SASM14	\$YFF470	xx001110	14
RTCSM16	\$YFF480	xx010000	16
PIOSM17A	\$YFF488	—	—
SASM18	\$YFF490	xx010010	18
SASM24	\$YFF4C0	xx011000	24
DASM26	\$YFF4D0	xx011010	26
DASM27	\$YFF4D8	xx011011	27
DASM28	\$YFF4E0	xx011100	28
DASM29	\$YFF4E8	xx011101	29
MCSM30	\$YFF4F0	xx011110	30
MCSM31	\$YFF4F8	xx011111	31
RAMSM32	\$YFF500	—	—
RAMSM36	\$YFF520	—	—

**NOTES:**

1. Y = m111, where m is the state of the MM bit in SIMLCR of the SIML (Y = \$7 or \$F).
2. "xx" represents VECT[7:6], which is located in the BIUSM module configuration register.
3. Submodule interrupt arbitration number 2 is the highest priority; arbitration number 63 is the lowest priority.

## 7 Electrical Characteristics

This section contains electrical specification tables and reference timing diagrams.

**Table 67 Maximum Ratings**

Num	Rating	Symbol	Value	Unit
1	Supply Voltage <sup>1, 2, 3</sup>	$V_{DD}$	- 0.3 to + 6.5	V
2	Input Voltage <sup>1, 2, 3, 4</sup>	$V_{in}$	- 0.3 to + 6.5	V
3	Instantaneous Maximum Current Single pin limit (applies to all pins) <sup>1, 3, 5, 6</sup>	$I_D$	25	mA
4	Operating Maximum Current Digital Input Disruptive Current <sup>5, 6, 7</sup> $V_{SS} - 0.3 \leq V_{IN} \leq V_{DD} + 0.3$	$I_{ID}$	- 500 to + 500	$\mu$ A
5	Operating Temperature Range	$T_A$	$T_L$ to $T_H$ - 40 to + 85	$^{\circ}$ C
6	Storage Temperature Range	$T_{stg}$	- 55 to + 150	$^{\circ}$ C

**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.
3. This parameter is periodically sampled rather than 100% tested.
4. All pins except TSC.
5. All functional non-supply pins are internally clamped to  $V_{SS}$  for transitions below  $V_{SS}$ . All functional pins except EXTAL and XFC are internally clamped to  $V_{DD}$  for transitions below  $V_{DD}$ .
6. Power supply must maintain regulation within operating  $V_{DD}$  range during instantaneous and operating maximum current conditions.
7. 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.

**Table 68 Thermal Characteristics**

Num	Characteristic	Symbol	Value	Unit
1	Thermal Resistance Plastic 144-Pin Surface Mount	$\Theta_{JA}$	48	$^{\circ}\text{C}/\text{W}$

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,  $^{\circ}\text{C}$
- $\Theta_{JA}$  = Package Thermal Resistance, Junction-to-Ambient,  $^{\circ}\text{C}/\text{W}$
- $P_D$  =  $P_{INT} + P_{I/O}$
- $P_{INT}$  =  $I_{DD} \times V_{DD}$ , Watts — Chip Internal Power
- $P_{I/O}$  = Power Dissipation on Input and Output Pins — User Determined

For most applications  $P_{I/O} < P_{INT}$  and can be neglected. An approximate relationship between  $P_D$  and  $T_J$  (if  $P_{I/O}$  is neglected) is:

$$P_D = K \div (T_J + 273^{\circ}\text{C}) \quad (2)$$

Solving equations 1 and 2 for K gives:

$$K = P_D + (T_A + 273^{\circ}\text{C}) + \Theta_{JA} \times P_D^2 \quad (3)$$

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 69 Clock Control Timing** $(V_{DD}$  and  $V_{DDSYN} = 2.7$  to  $3.6$  Vdc,  $V_{SS} = 0$  Vdc,  $T_A = T_L$  to  $T_H$ )

Num	Characteristic	Symbol	Min	Max	Unit
1	PLL Reference Frequency Range <sup>1</sup>	$f_{ref}$	25	50	kHz
2	System Frequency <sup>2</sup> On-Chip PLL System Frequency Range External Clock Operation	$f_{sys}$	dc 4( $f_{ref}$ ) dc	14.4 14.4 14.4	MHz
3	PLL Lock Time <sup>1, 3, 4, 5, 6</sup>	$t_{lpll}$	—	20	ms
4	VCO Frequency <sup>7</sup>	$f_{VCO}$	—	2 ( $f_{sys}$ max)	MHz
5	Limp Mode Clock Frequency SYNCR X bit = 0 SYNCR X bit = 1	$f_{limp}$	— —	$f_{sys}$ max/2 $f_{sys}$ max	MHz
6	CLKOUT Jitter <sup>1, 4, 5, 6, 8</sup> Short term (5 $\mu$ s interval) Long term (500 $\mu$ s interval)	$J_{clk}$	-0.5 -0.05	0.5 0.05	%

**NOTES:**

1. Tested with a 32.768 kHz reference.
2. All internal registers retain data at 0 Hz.
3. Assumes that stable  $V_{DDSYN}$  is applied, and that the crystal oscillator is stable. Lock time is measured from the time  $V_{DD}$  and  $V_{DDSYN}$  are valid until **RESET** is released. This specification also applies to the period required for PLL lock after changing the W and Y frequency control bits in the synthesizer control register (SYNCR) while the PLL is running, and to the period required for the clock to lock after LPSTOP.
4. This parameter is periodically sampled rather than 100% tested.
5. Assumes that a low-leakage external filter network is used to condition clock synthesizer input voltage. Total external resistance from the XFC pin due to external leakage must be greater than 15 M $\Omega$  to guarantee this specification. Filter network geometry can vary depending upon operating environment.
6. Proper layout procedures must be followed to achieve specifications.
7. Internal VCO frequency ( $f_{VCO}$ ) is determined by SYNCR W and Y bit values.  
The SYNCR X bit controls a divide-by-two circuit that is not in the synthesizer feedback loop.  
When X = 0, the divider is enabled, and  $f_{sys} = f_{VCO} \div 4$ .  
When X = 1, the divider is disabled, and  $f_{sys} = f_{VCO} \div 2$ .  
X must equal one when operating at maximum specified  $f_{sys}$ .
8. 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  $V_{DDSYN}$  and  $V_{SS}$  and variation in crystal oscillator frequency increase the  $J_{clk}$  percentage for a given interval. When clock jitter is a critical constraint on control system operation, this parameter should be measured during functional testing of the final system.

**Table 70 DC Characteristics**
 $(V_{DD} \text{ and } V_{DDSYN} = 2.7 \text{ to } 3.6 \text{ Vdc, } V_{SS} = 0 \text{ Vdc, } T_A = T_L \text{ to } T_H)$ 

Num	Characteristic	Symbol	Min	Max	Unit
1	Input High Voltage	$V_{IH}$	0.7 ( $V_{DD}$ )	$V_{DD} + 0.3$	V
2	Input Low Voltage	$V_{IL}$	$V_{SS} - 0.3$	0.2 ( $V_{DD}$ )	V
3	Input Hysteresis <sup>1</sup>	$V_{HYS}$	0.5	—	V
4	Input Leakage Current <sup>2</sup> $V_{in} = V_{DD}$ or $V_{SS}$ Input-only pins	$I_{IN}$	-2.5	2.5	$\mu\text{A}$
5	High Impedance (Off-State) Leakage Current <sup>2</sup> $V_{in} = V_{DD}$ or $V_{SS}$ All input/output and output pins	$I_{OZ}$	-2.5	2.5	$\mu\text{A}$
6	CMOS Output High Voltage <sup>2, 3</sup> $I_{OH} = -10.0 \mu\text{A}$ Group 1, 2, 4 input/output and all output pins	$V_{OH}$	$V_{DD} - 0.2$	—	V
7	CMOS Output Low Voltage <sup>2</sup> $I_{OL} = 10.0 \mu\text{A}$ Group 1, 2, 4 input/output and all output pins	$V_{OL}$	—	0.2	V
8	Output High Voltage <sup>2, 3</sup> $I_{OH} = -0.4 \text{ mA}$ Group 1, 2, 4 input/output and all output pins	$V_{OH}$	$V_{DD} - 0.5$	—	V
9	Output Low Voltage <sup>2</sup> $I_{OL} = 0.8 \text{ mA}$ Group 1 I/O Pins, CLKOUT, FREEZE/QUOT, IPIPE/DSO $I_{OL} = 2.65 \text{ mA}$ Group 2 and Group 4 I/O Pins, CSBOOT, BG/CS1 $I_{OL} = 6 \text{ mA}$ Group 3	$V_{OL}$	— — —	0.4 0.4 0.4	V
10	Three State Control Input High Voltage	$V_{IHTSC}$	2.7 ( $V_{DD}$ )	9.1	V
11	Data Bus Mode Select Pull-up Current <sup>4</sup> $V_{in} = V_{IL}$ DATA[15:0] $V_{in} = V_{IH}$ DATA[15:0]	$I_{MSP}$	— -8	-95 —	$\mu\text{A}$
12	$V_{DD}$ Supply Current <sup>5</sup> Run <sup>6</sup> LPSTOP, (STCPU = 0, External clock input frequency = max $f_{sys}$ ) LPSTOP, (STCPU = 1, External clock input frequency = max $f_{sys}$ ) STOP <sup>7</sup> , (STCPU = 1, External clock input frequency = max $f_{sys}$ )	$I_{DD}$ $S_{IDD}$ $S_{IDD}$ $S_{IDD}$	— — — —	46 2 18 13	 mA mA mA mA
13	$V_{RTC}$ Voltage	$V_{SB}$	2.2	3.6	V
14	$V_{RTC}$ Current <sup>8</sup> CTM6-RTCSM oscillator enabled, $V_{DD} = V_{SS}$ CTM6-RTCSM oscillator disabled, $V_{DD} = V_{SS}$ $V_{DD} = V_{DDSYN} \geq 2.7 \text{ V}$	$I_{SB}$	— — —	1.0 0.200 0	$\mu\text{A}$ $\mu\text{A}$ $\mu\text{A}$
15	$V_{DDSYN}$ Supply Current <sup>5</sup> VCO on, 32.768 kHz crystal reference, maximum $f_{sys}$ External Clock, maximum $f_{sys}$ LPSTOP, 32.768 kHz crystal reference, VCO off (STSIM = 0)	$I_{DDSYN}$	— — —	750 1.5 300	$\mu\text{A}$ mA $\mu\text{A}$
16	Power Dissipation <sup>9</sup>	$P_D$	—	171	mW

**Table 70 DC Characteristics (Continued)**

( $V_{DD}$  and  $V_{DDSYN} = 2.7$  to  $3.6$  Vdc,  $V_{SS} = 0$  Vdc,  $T_A = T_L$  to  $T_H$ )

Num	Characteristic	Symbol	Min	Max	Unit
17	Input Capacitance <sup>2, 10</sup> All input-only pins All input/output pins	$C_{IN}$	— —	10 20	pF
18	Load Capacitance <sup>2</sup> Group 1 I/O Pins, CLKOUT, FREEZE/QUOT, $\overline{IPIPE}/DSO$ Group 2 I/O Pins and $\overline{CSBOOT}$ , $\overline{BG}/\overline{CS1}$ Group 3 I/O Pins Group 4 I/O Pins	$C_L$	— — — —	90 100 130 200	pF

NOTES:

- Applies to:  
CTM6 pins  
QSM pins  
 $\overline{IRQ}[7:1]$ ,  $\overline{RESET}$ ,  $\overline{EXTAL}$ ,  $\overline{TSC}$ ,  $\overline{RMC}$ ,  $\overline{BKPT}/\overline{DSCLK}$ ,  $\overline{IFETCH}/\overline{DSI}$
- Input-Only Pins:  $\overline{TSC}$ ,  $\overline{BKPT}/\overline{DSCLK}$ ,  $\overline{RXD}$   
Output-Only Pins:  $\overline{CSBOOT}$ ,  $\overline{BG}/\overline{CS1}$ ,  $\overline{CLKOUT}$ ,  $\overline{FREEZE}/\overline{QUOT}$ ,  $\overline{IPIPE}/\overline{DSO}$   
Input/Output Pins:  
Group 1:  $\overline{DATA}[15:0]$ ,  $\overline{IFETCH}/\overline{DSI}$ , all CTM6 pins except CTM31L  
Group 2:  $\overline{ADDR}[23:19]/\overline{CS}[10:6]$ ,  $\overline{FC}[2:0]/\overline{CS}[5:3]$ ,  $\overline{DSACK}[1:0]$ ,  $\overline{AVEC}$ ,  $\overline{RMC}$ ,  $\overline{DS}$ ,  $\overline{AS}$ ,  $\overline{SIZ}[1:0]$ ,  $\overline{IRQ}[7:1]$ ,  $\overline{MODCLK}$ ,  $\overline{ADDR}[18:0]$ ,  $\overline{R/W}$ ,  $\overline{BERR}$ ,  $\overline{BR}/\overline{CS0}$ ,  $\overline{BGACK}/\overline{CS2}$ ,  $\overline{PCS}[3:1]$ ,  $\overline{PCS0}/\overline{SS}$ ,  $\overline{TXD}$   
Group 3:  $\overline{HALT}$ ,  $\overline{RESET}$   
Group 4:  $\overline{MISO}$ ,  $\overline{MOSI}$ ,  $\overline{SCK}$
- Does not apply to  $\overline{HALT}$  and  $\overline{RESET}$  because they are open drain pins.  
Does not apply to  $\overline{MISO}$ ,  $\overline{MOSI}$ ,  $\overline{SCK}$ ,  $\overline{PCS0}/\overline{SS}$ ,  $\overline{PCS}[3:1]$ , and  $\overline{TXD}$  in wired-OR mode.  
Does not apply to  $\overline{CTD}[29:26]$  and  $\overline{CTD}[10:4]$  in wired-OR mode.
- Use of an active pulldown device is recommended.
- Total operating current is the sum of the appropriate  $V_{DD}$  supply and  $V_{DDSYN}$  supply current.
- Current measured with system clock frequency of 14.4 MHz, all modules active.
- LPSTOP with  $\overline{STCPU} = 1$  (clock turned off at CPU32L but IMB clock active) plus QSM and CTM6 STOP bits set.
- $V_{RTC}$  current measured when  $V_{DD}$  and  $V_{DDSYN}$  are equal to  $V_{SS}$  and  $V_{RTC}$  is equal to  $V_{SBMAX}$ .
- Power dissipation is measured with a system clock frequency of 14.4 MHz, all modules active. Power dissipation is calculated using the following expression:  
$$P_D = 3.6V (I_{DDSYN} + I_{DD})$$
- Input capacitance is periodically sampled rather than 100% tested.

**Table 71 AC Timing** $(V_{DD} \text{ and } V_{DDSYN} = 2.7 \text{ to } 3.6 \text{ Vdc}, V_{SS} = 0 \text{ Vdc}, T_A = T_L \text{ to } T_H)^1$ 

Num	Characteristic	Symbol	Min	Max	Unit
F1	Frequency of Operation	f	0	14.4	MHz
1	Clock Period	$t_{cyc}$	69.4	—	ns
1A	ECLK Period	$t_{Ecyc}$	555	—	ns
1B	External Clock Input Period <sup>2</sup>	$t_{xcyc}$	69.4	—	ns
2, 3	Clock Pulse Width	$t_{cw}$	24.7	—	ns
2A, 3A	ECLK Pulse Width	$t_{ECW}$	277.5	—	ns
2B, 3B	External Clock Input High/Low Time <sup>2</sup>	$t_{xchl}$	34.7	—	ns
4, 5	CLKOUT Rise and Fall Time	$t_{crf}$	—	10	ns
4A, 5A	Rise and Fall Time — All Outputs Except CLKOUT	$t_{rf}$	—	10	ns
6	Clock High to ADDR, FC, SIZE, $\overline{RMC}$ Valid	$t_{CHAV}$	0	35	ns
7	Clock High to ADDR, DATA, FC, SIZE, $\overline{RMC}$ High Impedance	$t_{CHAZx}$	0	69	ns
8	Clock High to ADDR, FC, SIZE, $\overline{RMC}$ Invalid	$t_{CHAZn}$	0	—	ns
9	Clock Low to $\overline{AS}$ , $\overline{DS}$ , $\overline{CS}$ Asserted	$t_{CLSA}$	2	25	ns
9A	$\overline{AS}$ to $\overline{DS}$ or $\overline{CS}$ Asserted (Read) <sup>3</sup>	$t_{STSA}$	-15	15	ns
9C	Clock Low to $\overline{IFETCH}$ , $\overline{IPIPE}$ Asserted	$t_{CLIA}$	2	31	ns
11	ADDR, FC, SIZE, $\overline{RMC}$ Valid to $\overline{AS}$ , $\overline{CS}$ , (and $\overline{DS}$ Read) Asserted	$t_{AVSA}$	15	—	ns
12	Clock Low to $\overline{AS}$ , $\overline{DS}$ , $\overline{CS}$ Negated	$t_{CLSN}$	2	35	ns
12A	Clock Low to $\overline{IFETCH}$ , $\overline{IPIPE}$ Negated	$t_{CLIN}$	2	31	ns
13	$\overline{AS}$ , $\overline{DS}$ , $\overline{CS}$ Negated to ADDR, FC SIZE Invalid (Address Hold)	$t_{SNAI}$	19	—	ns
14	$\overline{AS}$ , $\overline{CS}$ (and $\overline{DS}$ Read) Width Asserted	$t_{SWA}$	138	—	ns
14A	$\overline{DS}$ , $\overline{CS}$ Width Asserted (Write)	$t_{SWAW}$	44	—	ns
14B	$\overline{AS}$ , $\overline{CS}$ (and $\overline{DS}$ Read) Width Asserted (Fast Cycle)	$t_{SWDW}$	44	—	ns
15	$\overline{AS}$ , $\overline{DS}$ , $\overline{CS}$ Width Negated <sup>4</sup>	$t_{SN}$	44	—	ns
16	Clock High to $\overline{AS}$ , $\overline{DS}$ , $R/\overline{W}$ High Impedance	$t_{CHSZ}$	—	69	ns
17	$\overline{AS}$ , $\overline{DS}$ , $\overline{CS}$ Negated to $R/\overline{W}$ High	$t_{SNRN}$	19	—	ns
18	Clock High to $R/\overline{W}$ High	$t_{CHRH}$	0	35	ns
20	Clock High to $R/\overline{W}$ Low	$t_{CHRL}$	0	35	ns
21	$R/\overline{W}$ High to $\overline{AS}$ , $\overline{CS}$ Asserted	$t_{RAAA}$	19	—	ns
22	$R/\overline{W}$ Low to $\overline{DS}$ , $\overline{CS}$ Asserted (Write)	$t_{RASA}$	80	—	ns
23	Clock High to Data Out Valid	$t_{CHDO}$	—	35	ns
24	Data Out Valid to Negating Edge of $\overline{AS}$ , $\overline{CS}$ (Fast Write Cycle)	$t_{DVASN}$	19	—	ns
25	$\overline{DS}$ , $\overline{CS}$ Negated to Data Out Invalid (Data Out Hold)	$t_{SNDOI}$	19	—	ns
26	Data Out Valid to $\overline{DS}$ , $\overline{CS}$ Asserted (Write)	$t_{DVSA}$	19	—	ns

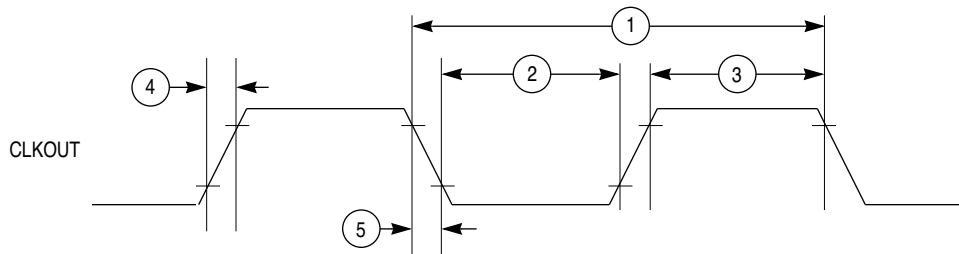
**Table 71 AC Timing (Continued)**

( $V_{DD}$  and  $V_{DSDYN} = 2.7$  to  $3.6$  Vdc,  $V_{SS} = 0$  Vdc,  $T_A = T_L$  to  $T_H$ )<sup>1</sup>

Num	Characteristic	Symbol	Min	Max	Unit
27	Data In Valid to Clock Low (Data Setup)	$t_{DICL}$	5	—	ns
27A	Late $\overline{BERR}$ , $\overline{HALT}$ Asserted to Clock Low (Setup Time)	$t_{BELCL}$	25	—	ns
28	$\overline{AS}$ , $\overline{DS}$ Negated to $\overline{DSACK}[1:0]$ , $\overline{BERR}$ , $\overline{HALT}$ , $\overline{AVEC}$ Negated	$t_{SNDN}$	0	100	ns
29	$\overline{DS}$ , $\overline{CS}$ Negated to Data In Invalid (Data In Hold) <sup>5</sup>	$t_{SNDI}$	0	—	ns
29A	$\overline{DS}$ , $\overline{CS}$ Negated to Data In High Impedance <sup>5, 6</sup>	$t_{SHDI}$	—	65	ns
30	CLKOUT Low to Data In Invalid (Fast Cycle Hold) <sup>5</sup>	$t_{CLDI}$	19	—	ns
30A	CLKOUT Low to Data In High Impedance <sup>5</sup>	$t_{CLDH}$	—	100	ns
31	$\overline{DSACK}[1:0]$ Asserted to Data In Valid <sup>7</sup>	$t_{DADI}$	—	58	ns
33	Clock Low to $\overline{BG}$ Asserted/Negated	$t_{CLBAN}$	—	35	ns
35	$\overline{BR}$ Asserted to $\overline{BG}$ Asserted ( $\overline{RMC}$ Not Asserted) <sup>8</sup>	$t_{BRAGA}$	1	—	$t_{cyc}$
37	$\overline{BGACK}$ Asserted to $\overline{BG}$ Negated	$t_{GAGN}$	1	2	$t_{cyc}$
39	$\overline{BG}$ Width Negated	$t_{GH}$	2	—	$t_{cyc}$
39A	$\overline{BG}$ Width Asserted	$t_{GA}$	1	—	$t_{cyc}$
46	$R/\overline{W}$ Width Asserted (Write or Read)	$t_{RWA}$	174	—	ns
46A	$R/\overline{W}$ Width Asserted (Fast Write or Read Cycle)	$t_{RWAS}$	104	—	ns
47A	Asynchronous Input Setup Time $\overline{BR}$ , $\overline{BGACK}$ , $\overline{DSACK}[1:0]$ , $\overline{BERR}$ , $\overline{AVEC}$ , $\overline{HALT}$	$t_{AIST}$	10	—	ns
47B	Asynchronous Input Hold Time	$t_{AIHT}$	19	—	ns
48	$\overline{DSACK}[1:0]$ Asserted to $\overline{BERR}$ , $\overline{HALT}$ Asserted <sup>9</sup>	$t_{DABA}$	—	35	ns
53	Data Out Hold from Clock High	$t_{DOCH}$	0	—	ns
54	Clock High to Data Out High Impedance	$t_{CHDH}$	—	35	ns
55	$R/\overline{W}$ Asserted to Data Bus Impedance Change	$t_{RADC}$	55	—	ns
56	RESET Pulse Width (Reset Instruction)	$t_{HRPW}$	512	—	$t_{cyc}$
57	$\overline{BERR}$ Negated to $\overline{HALT}$ Negated (Rerun)	$t_{BNHN}$	0	—	ns
70	Clock Low to Data Bus Driven (Show Cycle)	$t_{SCLDD}$	0	35	ns
71	Data Setup Time to Clock Low (Show Cycle)	$t_{SCLDS}$	19	—	ns
72	Data Hold from Clock Low (Show Cycle)	$t_{SCLDH}$	10	—	ns
73	BKPT Input Setup Time	$t_{BKST}$	19	—	ns
74	BKPT Input Hold Time	$t_{BKHT}$	13	—	ns
75	Mode Select Setup Time (DATA[15:0], MODCLK, $\overline{BKPT}$ )	$t_{MSS}$	20	—	$t_{cyc}$
76	Mode Select Hold Time (DATA[15:0], MODCLK, $\overline{BKPT}$ )	$t_{MSH}$	0	—	ns
77	RESET Assertion Time <sup>10</sup>	$t_{RSTA}$	4	—	$t_{cyc}$
78	RESET Rise Time <sup>11, 12</sup>	$t_{RSTR}$	—	10	$t_{cyc}$

NOTES:

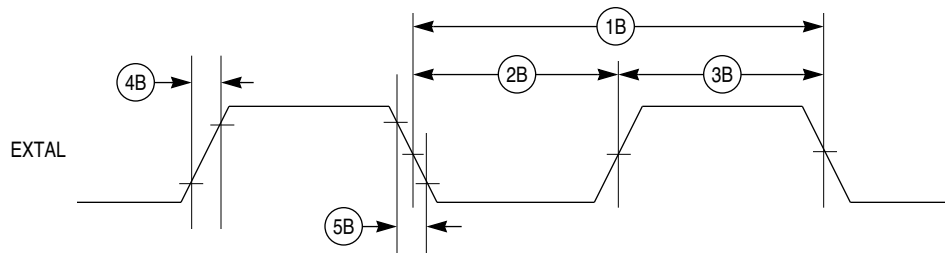
1. All AC timing is shown with respect to 20%  $V_{DD}$  and 70%  $V_{DD}$  levels unless otherwise noted.
2. When an external clock is used, minimum high and low times are based on a 50% duty cycle. The minimum allowable  $t_{X_{cyc}}$  period is reduced when the duty cycle of the external clock varies. The relationship between external clock input duty cycle and minimum  $t_{X_{cyc}}$  is expressed:  
Minimum  $t_{X_{cyc}}$  period = minimum  $t_{X_{CHL}}$  / (50% – external clock input duty cycle tolerance).
3. Specification 9A is the worst-case skew between  $\overline{AS}$  and  $\overline{DS}$  or  $\overline{CS}$ . The amount of skew depends on the relative loading of these signals. When loads are kept within specified limits, skew will not cause  $\overline{AS}$  and  $\overline{DS}$  to fall outside the limits shown in specification 9.
4. If multiple chip selects are used,  $\overline{CS}$  width negated (specification 15) applies to the time from the negation of a heavily loaded chip select to the assertion of a lightly loaded chip select. The  $\overline{CS}$  width negated specification between multiple chip selects does not apply to chip selects being used for synchronous ECLK cycles.
5. Hold times are specified with respect to  $\overline{DS}$  or  $\overline{CS}$  on asynchronous reads and with respect to CLKOUT on fast cycle reads. The user is free to use either hold time.
6. Maximum value is equal to  $(t_{cyc} / 2) + 25$  ns.
7. If the asynchronous setup time (specification 47A) requirements are satisfied, the  $\overline{DSACK}[1:0]$  low to data setup time (specification 31) and  $\overline{DSACK}[1:0]$  low to  $\overline{BERR}$  low setup time (specification 48) can be ignored. The data must only satisfy the data-in to clock low setup time (specification 27) for the following clock cycle.  $\overline{BERR}$  must satisfy only the late  $\overline{BERR}$  low to clock low setup time (specification 27A) for the following clock cycle.
8. To ensure coherency during every operand transfer,  $\overline{BG}$  is not asserted in response to  $\overline{BR}$  until after all cycles of the current operand transfer are complete.
9. In the absence of  $\overline{DSACK}[1:0]$ ,  $\overline{BERR}$  is an asynchronous input using the asynchronous setup time (specification 47A).
10. After external  $\overline{RESET}$  negation is detected, a short transition period (approximately  $2 t_{cyc}$ ) elapses, then the SIML drives  $\overline{RESET}$  low for  $512 t_{cyc}$ .
11. External assertion of the  $\overline{RESET}$  input can overlap internally-generated resets. To ensure that an external reset is recognized in all cases,  $\overline{RESET}$  must be asserted for at least 590 CLKOUT cycles.
12. External logic must pull  $\overline{RESET}$  high during this period in order for normal MCU operation to begin.



NOTE: TIMING SHOWN WITH RESPECT TO 20% AND 70%  $V_{DD}$ .

68300 CLKOUT TIM

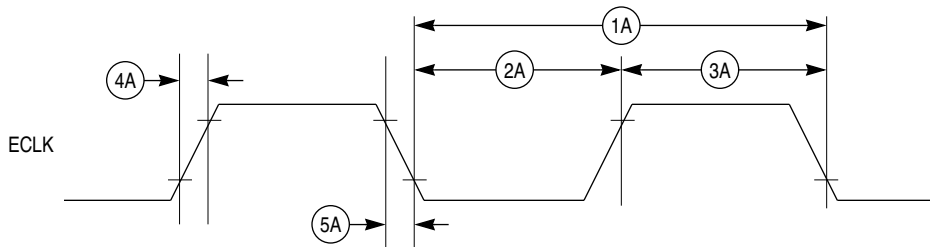
**Figure 26 CLKOUT Output Timing Diagram**



NOTE: TIMING SHOWN WITH RESPECT TO 20% AND 70%  $V_{DD}$ .  
PULSE WIDTH SHOWN WITH RESPECT TO 50%  $V_{DD}$ .

68300 EXT CLK INPUT TIM

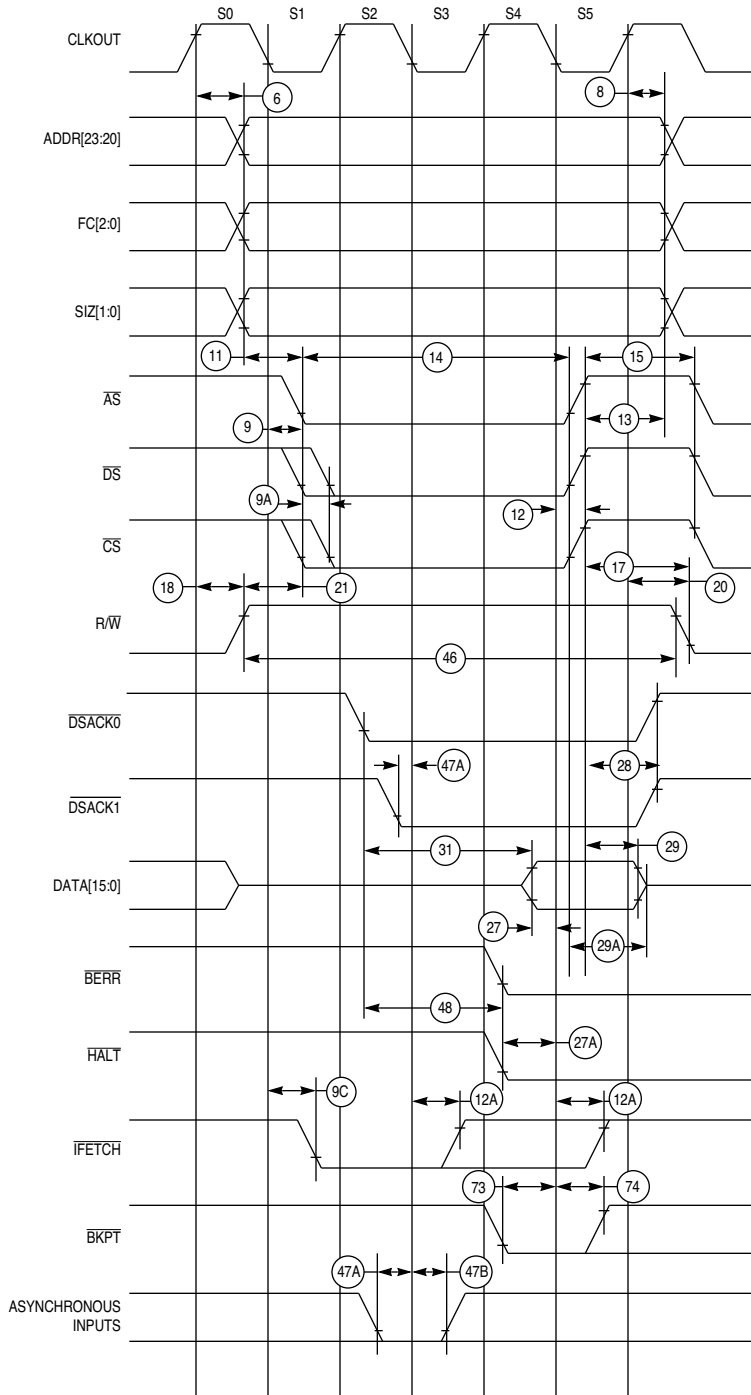
**Figure 27 External Clock Input Timing Diagram**



NOTE: TIMING SHOWN WITH RESPECT TO 20% AND 70%  $V_{DD}$ .

68300 ECLK OUTPUT TIM

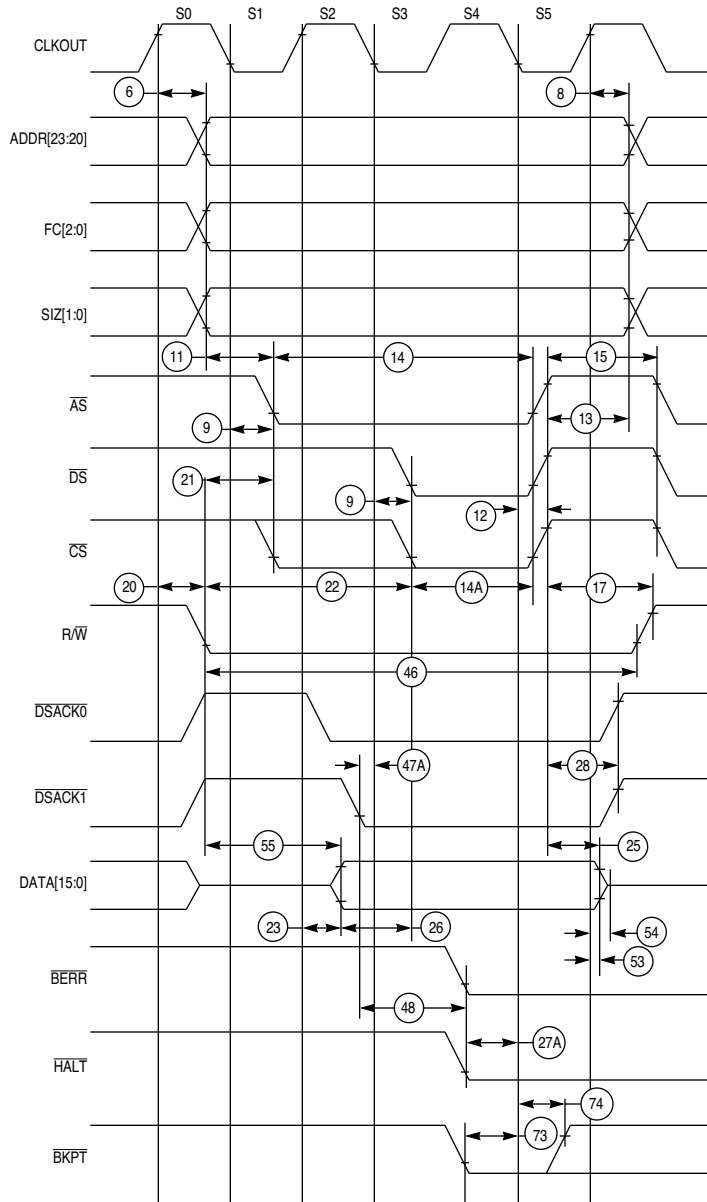
**Figure 28 ECLK Output Timing Diagram**



68300 RD CYC TIM

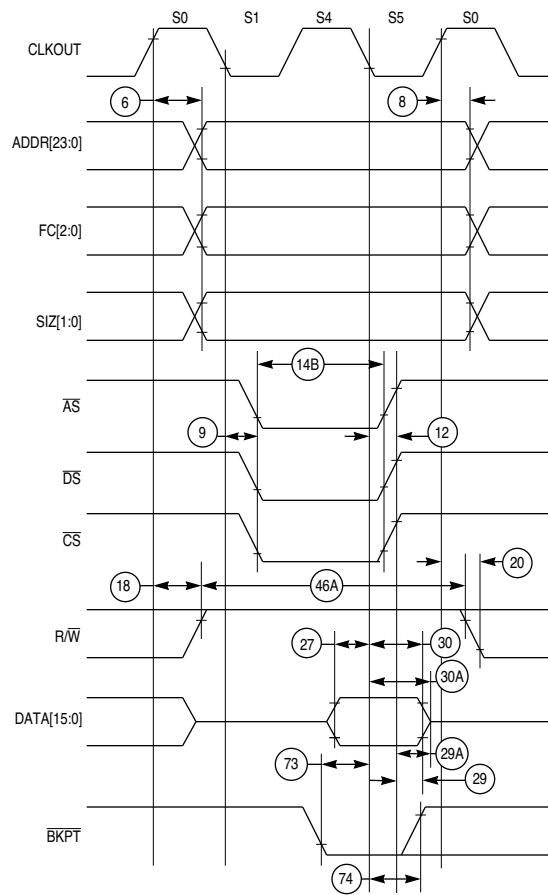
**Figure 29 Read Cycle Timing Diagram**





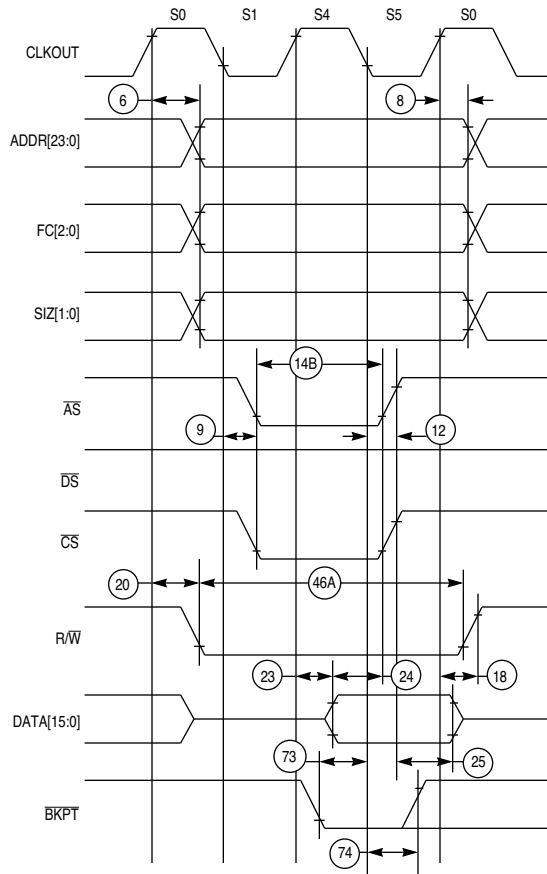
68300 WR CYC TIM

**Figure 30 Write Cycle Timing Diagram**



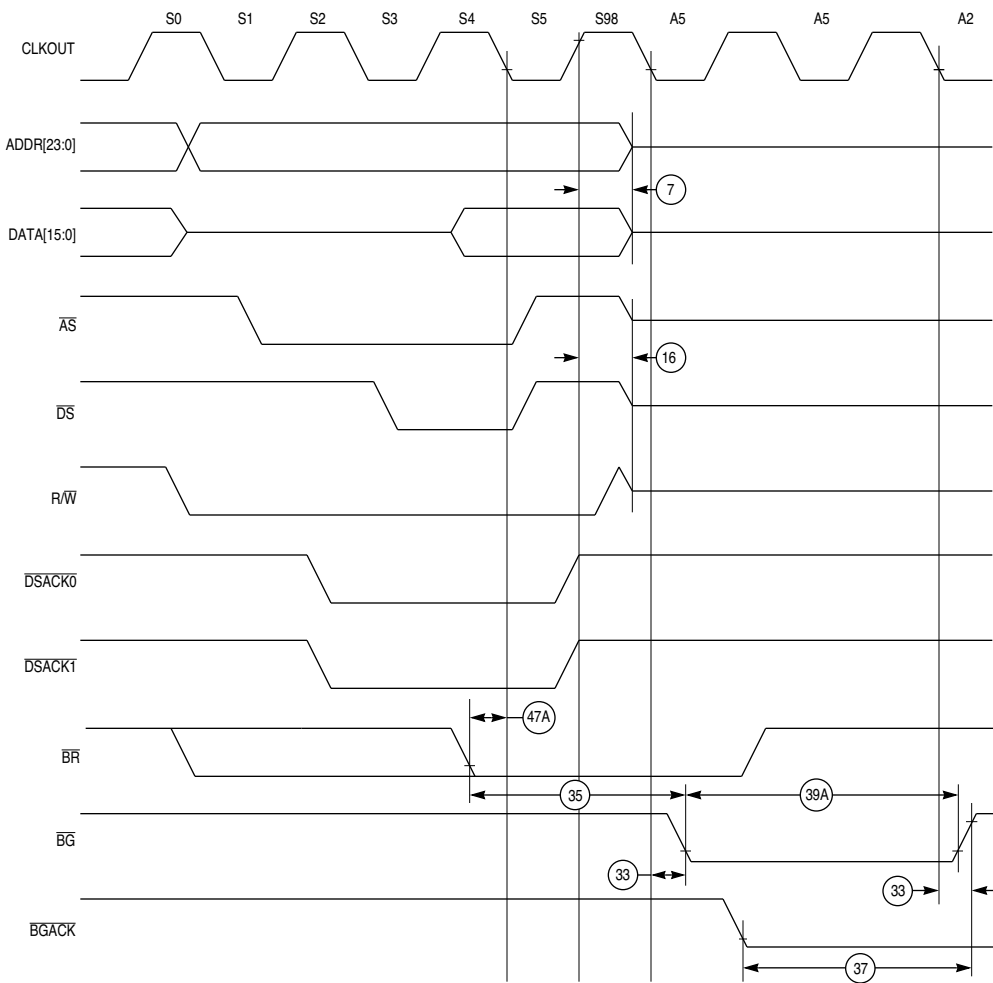
68300 FAST RD CYC TIM

**Figure 31 Fast Termination Read Cycle Timing Diagram**



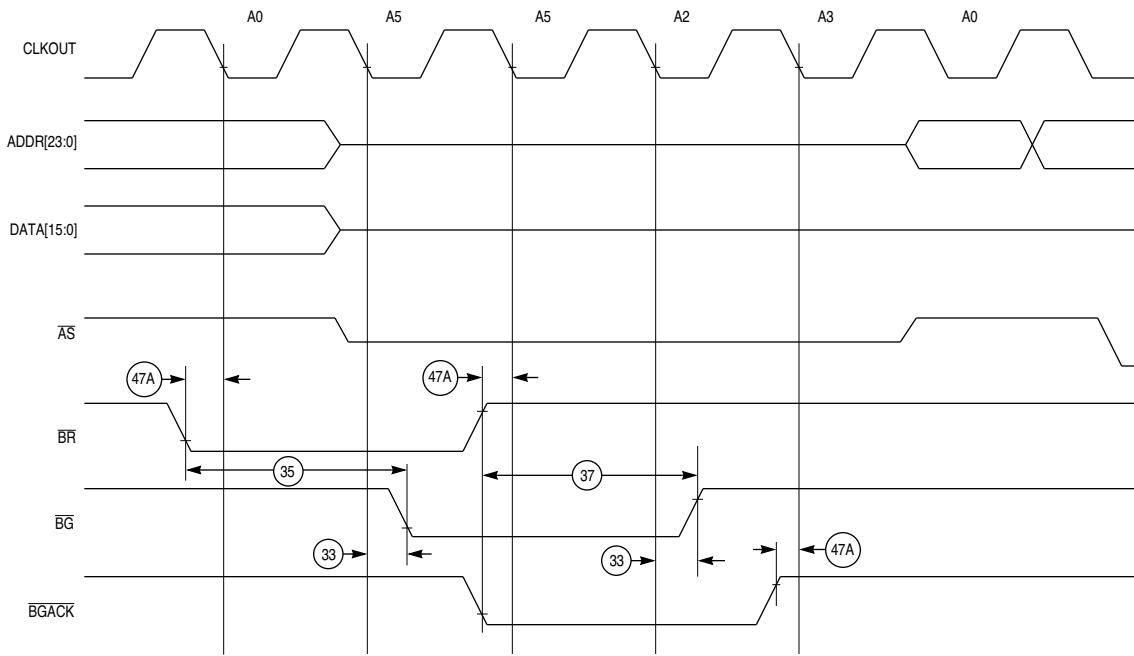
68300 FAST WR CYC TIM

**Figure 32 Fast Termination Write Cycle Timing Diagram**



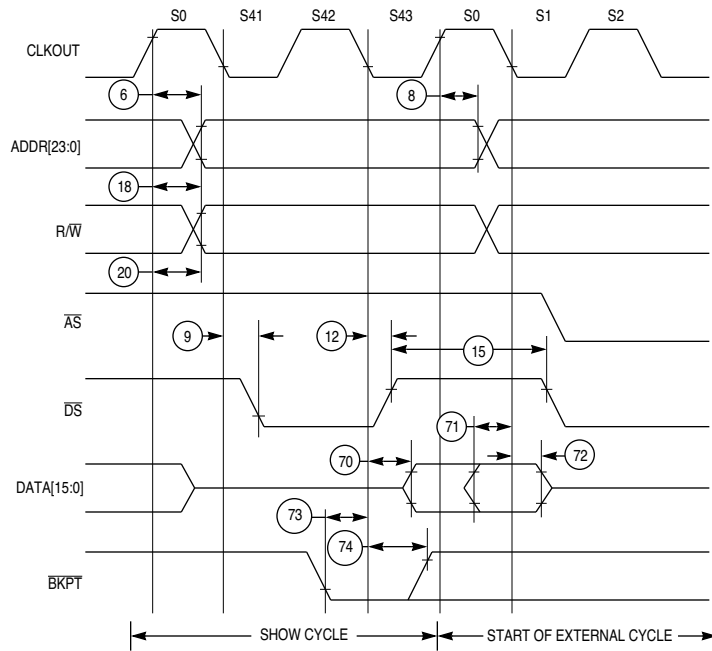
68300 BUS ARB TIM

**Figure 33 Bus Arbitration Timing Diagram — Active Bus Case**



68300 BUS ARB TIM IDLE

**Figure 34 Bus Arbitration Timing Diagram — Idle Bus Case**

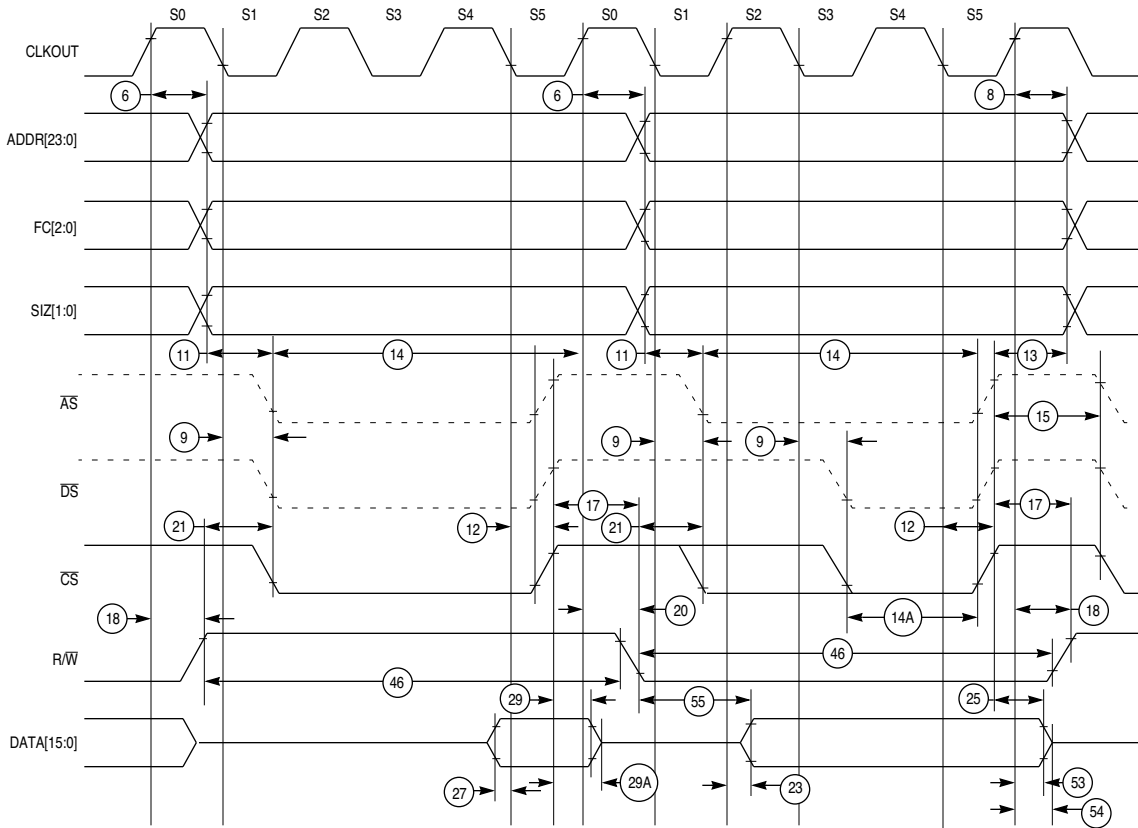


**NOTE:**

Show cycles can stretch during clock phase S42 when bus accesses take longer than two cycles due to IMB module wait-state insertion.

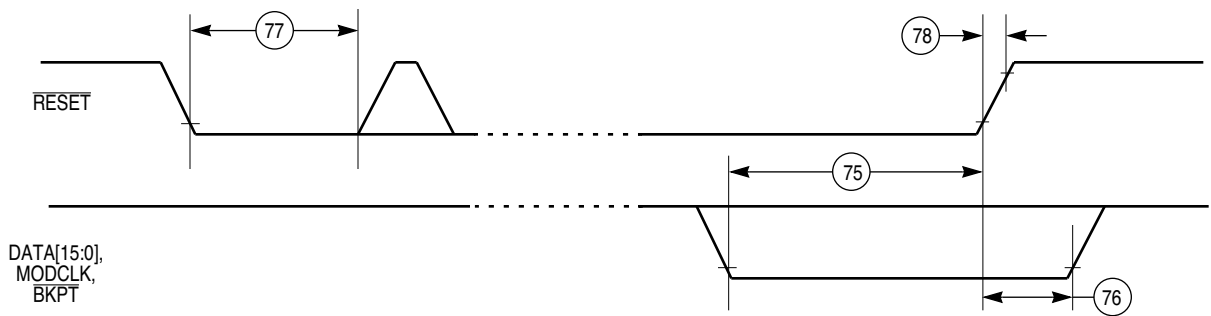
68300 SHW CYC TIM

**Figure 35 Show Cycle Timing Diagram**



68300 CHIP SEL TIM

**Figure 36 Chip Select Timing Diagram**



68300 RST/MODE SEL TIM

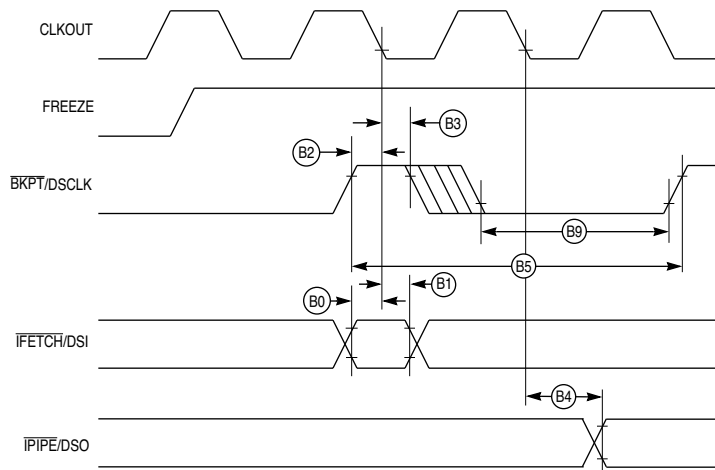
**Figure 37 Reset and Mode Select Timing Diagram**

**Table 72 Background Debugging Mode Timing**  
 $(V_{DD}$  and  $V_{DDSYN} = 2.7$  to  $3.6$  Vdc,  $V_{SS} = 0$  Vdc,  $T_A = T_L$  to  $T_H$ )<sup>1</sup>

Num	Characteristic	Symbol	Min	Max	Unit
B0	DSI Input Setup Time	$t_{DSISU}$	19	—	ns
B1	DSI Input Hold Time	$t_{DSIH}$	13	—	ns
B2	DSCLK Setup Time	$t_{DSCSU}$	19	—	ns
B3	DSCLK Hold Time	$t_{DSCH}$	13	—	ns
B4	DSO Delay Time	$t_{DSOD}$	—	35	ns
B5	DSCLK Cycle Time	$t_{DSCCYC}$	2	—	$t_{cyc}$
B6	CLKOUT High to FREEZE Asserted/Negated	$t_{FRZAN}$	—	64	ns
B7	CLKOUT High to $\overline{IFETCH}$ High Impedance	$t_{IFZ}$	—	64	ns
B8	CLKOUT High to $\overline{IFETCH}$ Valid	$t_{IF}$	—	64	ns
B9	DSCLK Low Time	$t_{DSCLO}$	1	—	$t_{cyc}$

**NOTES:**

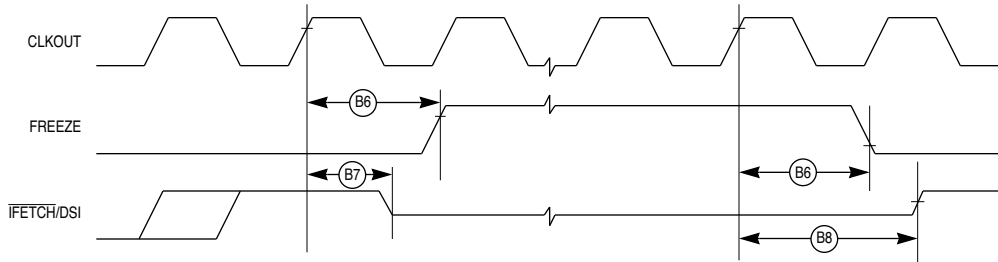
1. All AC timing is shown with respect to 20%  $V_{DD}$  and 70%  $V_{DD}$  levels unless otherwise noted.



68300 BKGD DBM SER COM TIM

**Figure 38 Background Debugging Mode Timing Diagram —  
Serial Communication**





68300 BDM FRZ TIM

**Figure 39 Background Debugging Mode Timing Diagram — Freeze Assertion**

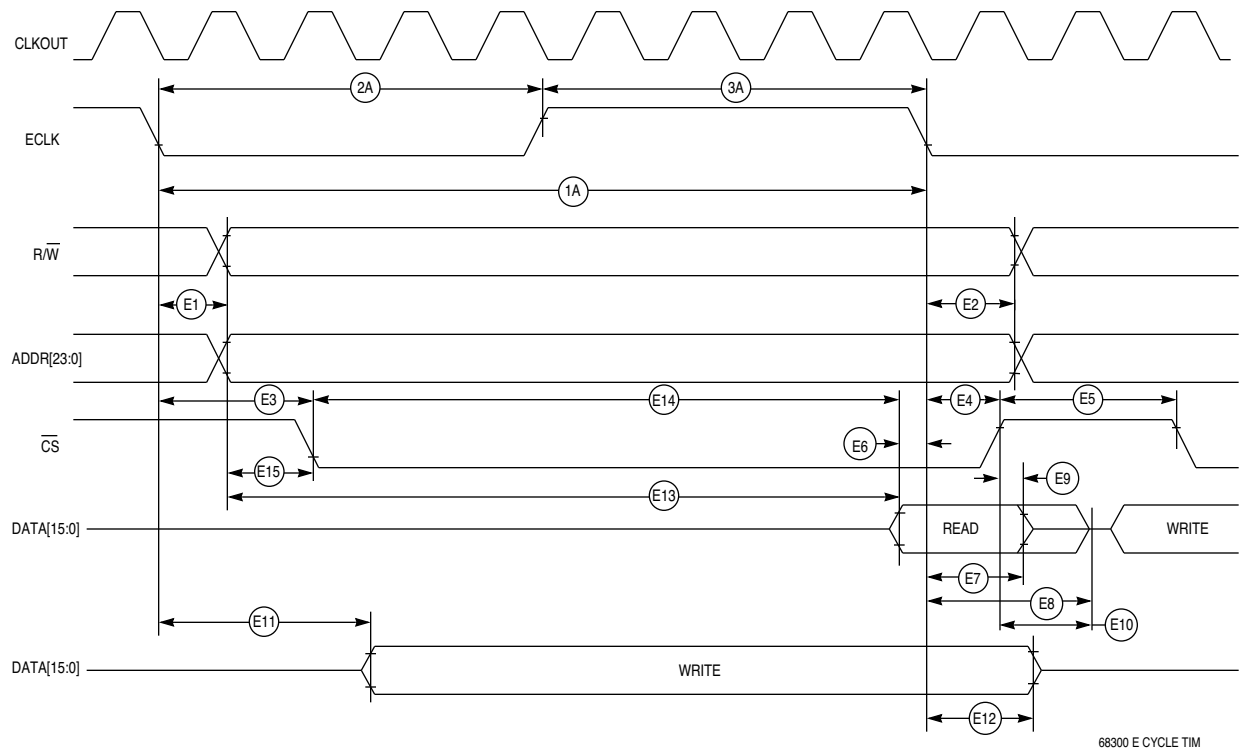
**Table 73 ECLK Bus Timing**

( $V_{DD}$  and  $V_{DDSYN} = 2.7$  to  $3.6$  Vdc,  $V_{SS} = 0$  Vdc,  $T_A = T_L$  to  $T_H$ )<sup>1</sup>

Num	Characteristic	Symbol	Min	Max	Unit
1A	ECLK Period	$t_{E\text{cyc}}$	555	—	ns
2A, 3A	ECLK Pulse Width	$t_{E\text{CW}}$	277.5	—	ns
E1	ECLK Low to Address Valid <sup>2</sup>	$t_{E\text{AD}}$	—	70	ns
E2	ECLK Low to Address Hold	$t_{E\text{AH}}$	10	—	ns
E3	ECLK Low to $\overline{\text{CS}}$ Valid ( $\overline{\text{CS}}$ delay)	$t_{E\text{CSD}}$	—	140	ns
E4	ECLK Low to $\overline{\text{CS}}$ Hold	$t_{E\text{CSH}}$	10	—	ns
E5	$\overline{\text{CS}}$ Negated Width	$t_{E\text{CSN}}$	35	—	ns
E6	Read Data Setup Time	$t_{E\text{DSR}}$	35	—	ns
E7	Read Data Hold Time	$t_{E\text{DHR}}$	5	—	ns
E8	ECLK Low to Data High Impedance	$t_{E\text{DHz}}$	—	130	ns
E9	$\overline{\text{CS}}$ Negated to Data Hold (Read)	$t_{E\text{CDH}}$	0	—	ns
E10	$\overline{\text{CS}}$ Negated to Data High Impedance	$t_{E\text{CDZ}}$	—	1	$t_{\text{cyc}}$
E11	ECLK Low to Data Valid (Write)	$t_{E\text{DDW}}$	—	2	$t_{\text{cyc}}$
E12	ECLK Low to Data Hold (Write)	$t_{E\text{DHW}}$	10	—	ns
E13	Address Access Time (Read) <sup>3</sup>	$t_{E\text{ACC}}$	450	—	ns
E14	Chip Select Access Time (Read) <sup>4</sup>	$t_{E\text{ACS}}$	380	—	ns
E15	Address Setup Time	$t_{E\text{AS}}$	1/2	—	$t_{\text{cyc}}$

**NOTES:**

1. All AC timing is shown with respect to 20%  $V_{DD}$  and 70%  $V_{DD}$  levels unless otherwise noted.
2. When the previous bus cycle is not an ECLK cycle, the address may be valid before ECLK goes low.
3. Address access time =  $t_{E\text{cyc}} - t_{E\text{AD}} - t_{E\text{DSR}}$ .
4. Chip select access time =  $t_{E\text{cyc}} - t_{E\text{CSD}} - t_{E\text{DSR}}$ .



**Figure 40 ECLK Timing Diagram**

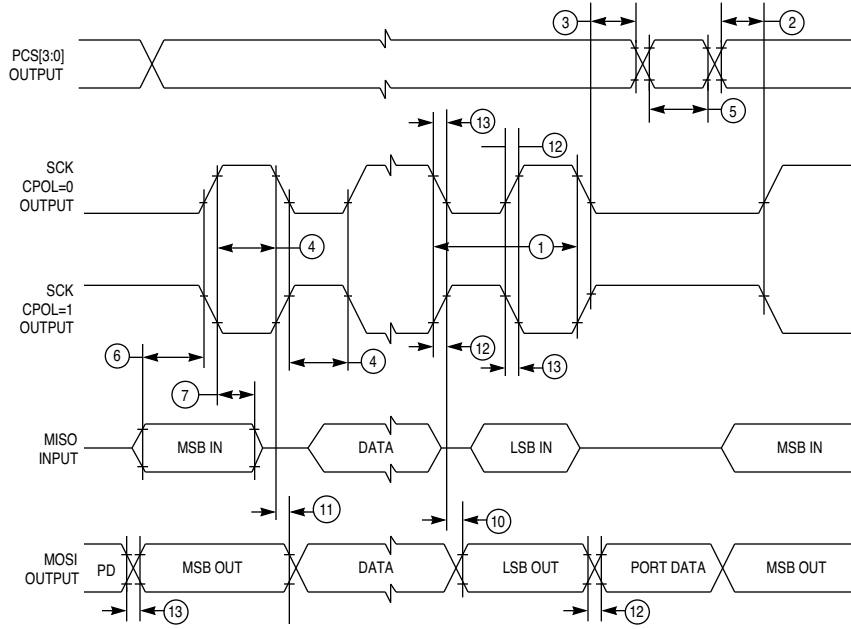
**Table 74 QSPI Timing**

( $V_{DD}$  and  $V_{DDSYN} = 2.7$  to  $3.6$  Vdc,  $V_{SS} = 0$  Vdc,  $T_A = T_L$  to  $T_H$  200 pF load on all QSPI pins)<sup>1</sup>

Num	Function	Symbol	Min	Max	Unit
1	Operating Frequency Master Slave	$f_{op}$	DC DC	1/4 1/4	System Clock Frequency System Clock Frequency
2	Cycle Time Master Slave	$t_{qcyt}$	4 4	510 —	$t_{cyc}$ $t_{cyc}$
3	Enable Lead Time Master Slave	$t_{lead}$	2 2	128 —	$t_{cyc}$ $t_{cyc}$
4	Enable Lag Time Master Slave	$t_{lag}$	— 2	1/2 —	SCK $t_{cyc}$
5	Clock (SCK) High or Low Time Master Slave <sup>2</sup>	$t_{sw}$	$2 t_{cyc} - 60$ $2 t_{cyc} - n$	$255 t_{cyc}$ —	ns ns
6	Sequential Transfer Delay Master Slave (Does Not Require Deselect)	$t_{td}$	17 13	8192 —	$t_{cyc}$ $t_{cyc}$
7	Data Setup Time (Inputs) Master Slave	$t_{su}$	45 30	— —	ns ns
8	Data Hold Time (Inputs) Master Slave	$t_{hi}$	0 30	— —	ns ns
9	Slave Access Time	$t_a$	—	1	$t_{cyc}$
10	Slave MISO Disable Time	$t_{dis}$	—	2	$t_{cyc}$
11	Data Valid (after SCK Edge) Master Slave	$t_v$	— —	75 75	ns ns
12	Data Hold Time (Outputs) Master Slave	$t_{ho}$	0 0	— —	ns ns
13	Rise Time Input Output	$t_{ri}$ $t_{ro}$	— —	2 45	$\mu$ s ns
14	Fall Time Input <sup>3</sup> Output	$t_{fi}$ $t_{fo}$	— —	2 45	$\mu$ s ns

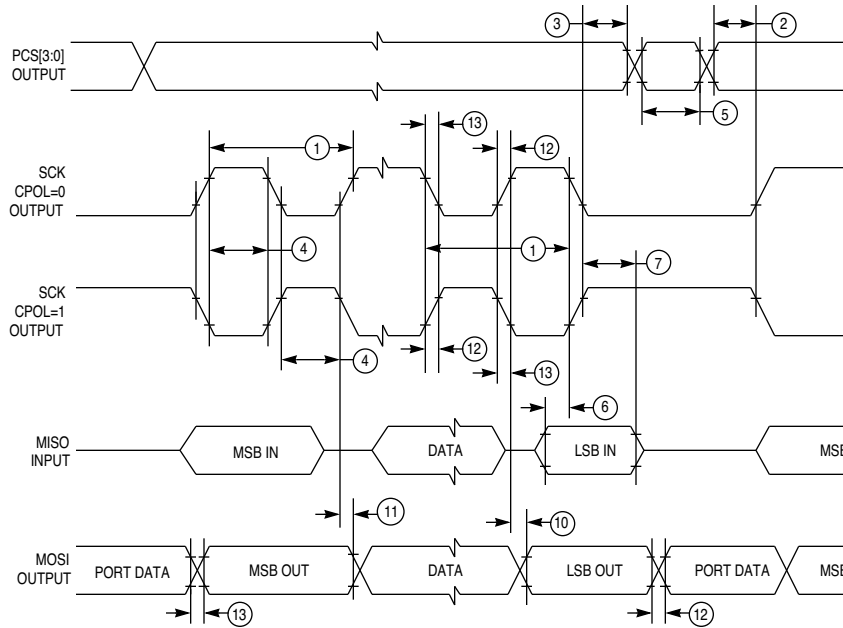
NOTES:

1. All AC timing is shown with respect to 20%  $V_{DD}$  and 70%  $V_{DD}$  levels unless otherwise noted.
2. For high time,  $n =$  External SCK rise time; for low time,  $n =$  External SCK fall time.
3. Data can be recognized properly with longer transition times as long as MOSI/MISO signals from external sources are at valid  $V_{OH}/V_{OL}$  prior to SCK transitioning between valid  $V_{OL}$  and  $V_{OH}$ . Due to process variation, logic decision point voltages of the data and clock signals can differ, which can corrupt data if slower transition times are used.



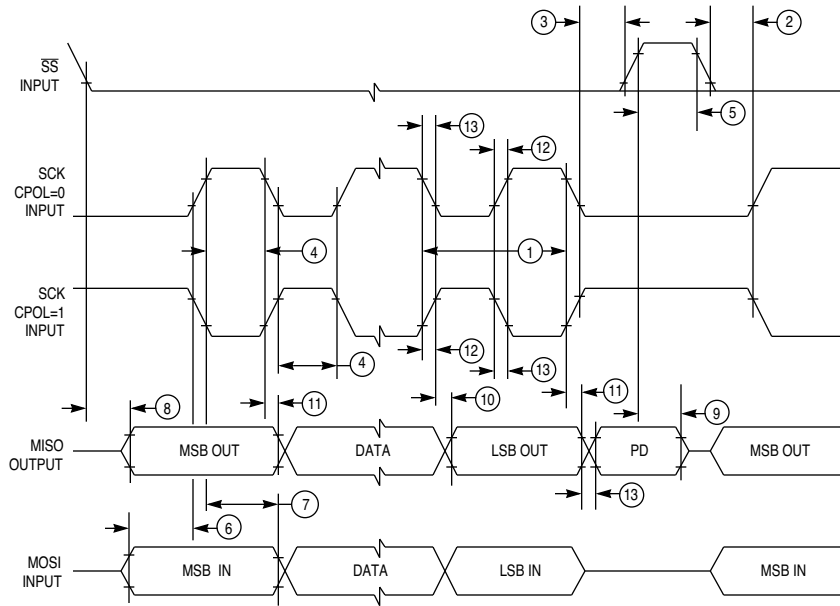
QSPI MAST CPHA0

**Figure 41 QSPI Timing — Master, CPHA = 0**



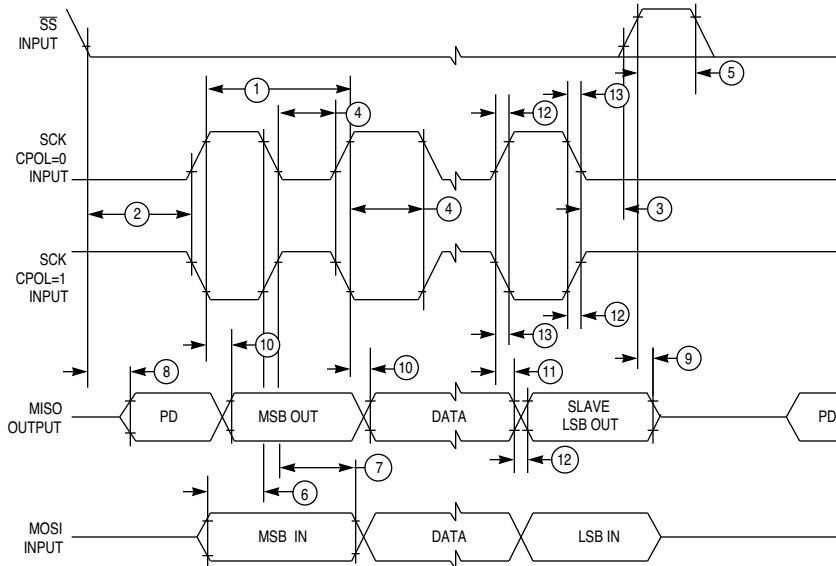
QSPI MAST CPHA1

**Figure 42 QSPI Timing — Master, CPHA = 1**



QSPI SLV CPHA0

**Figure 43 QSPI Timing — Slave, CPHA = 0**



QSPI SLV CPHA1

**Figure 44 QSPI Timing — Slave, CPHA = 1**