M68300 Family

MC68332

User's Manual

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Paragraph Title Page

TABLE OF CONTENTS
(Continued)

Paragraph Title Page

SECTION 5 CENTRAL PROCESSING UNIT

TABLE OF CONTENTS
(Continued)

Paragraph Title Page

SECTION 6QUEUED SERIAL MODULE

TABLE OF CONTENTS
(Continued)

SECTION 7TIME PROCESSOR UNIT

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

Paragraph Title Page

TABLE OF CONTENTS
(Continued)

Paragraph Title Page

TABLE OF CONTENTS
(Continued)

Paragraph Title Page

SUMMARY OF CHANGES

TABLE OF CONTENTS
(Continued)

Paragraph Title Page

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS
(Continued)

LIST OF TABLES

LIST OF TABLES
(Continued)

SECTION 1 INTRODUCTION

The MC68332, 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 32-bit CPU (CPU32), a system integration module (SIM), a time processor unit (TPU), a queued serial module (QSM), and a 2-Kbyte static RAM module with TPU emulation capability (TPURAM).

The MCU can either synthesize an internal clock signal from an external reference or use an external clock input directly. Operation with a 32.768-kHz reference frequency is standard. 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 makes the basic power consumption of the MCU low. Power consumption can be minimized by stopping the system clock. The CPU32 instruction set includes a low-power stop (LPSTOP) command that efficiently implements this capability.

Documentation for the Modular Microcontroller Family follows the modular construction of the devices in the product line. Each microcontroller has a comprehensive user's manual that provides sufficient information for normal operation of the device. The user's manual is supplemented by module reference manuals that provide detailed information about module operation and applications. Refer to Freescale publication *Advanced Microcontroller Unit (AMCU) Literature* (BR1116/D) for a complete listing of documentation.

SECTION 2 NOMENCLATURE

The following nomenclature is used throughout the manual. Nomenclature used only in certain sections, such as register bit mnemonics, is defined in those sections.

2.1 Symbols and Operators

+ — Addition
- — Subtraction or negation (two's complement)
* — Multiplication
/ — Division
> — Greater
< — Less
= — Equal
≥ — Equal or greater
≤ — Equal or less
- — Not equal
• — AND
; — Inclusive OR (OR)
⊕ — Exclusive OR (EOR)
$\overline{\text{NOT}}$ — Complementation
: — Concatenation
⇒ — Transferred
⇔ — Exchanged
± — Sign bit; also used to show tolerance
« — Sign extension
% — Binary value
$ — Hexadecimal value

2.2 CPU32 Registers

A6–A0 — Address registers (Index registers)
A7 (SSP) — Supervisor Stack Pointer
A7 (USP) — User Stack Pointer
CCR — Condition code register (user portion of SR)
D7–D0 — Data Registers (Index registers)
DFC — Alternate function code register
PC — Program counter
SFC — Alternate function code register
SR — Status register
VBR — Vector base register
X — Extend indicator
N — Negative indicator
Z — Zero indicator
V — Two's complement overflow indicator
C — Carry/borrow indicator

2.3 Pin and Signal Mnemonics

ADDR[23:0] — Address Bus
$\overline{AS}$ — Address Strobe
$\overline{AVEC}$ — Autovector
$\overline{BERR}$ — Bus Error
$\overline{BG}$ — Bus Grant
$\overline{BGACK}$ — Bus Grant Acknowledge
$\overline{BKPT}$ — Breakpoint
$\overline{BR}$ — Bus Request
CLKOUT — System Clock
$\overline{CS}$[10:0] — Chip Selects
$\overline{CSBOOT}$ — Boot ROM Chip Select
DATA[15:0] — Data Bus
$\overline{DS}$ — Data Strobe
$\overline{DSACK}$[1:0] — Data and Size Acknowledge
DSCLK — Development Serial Clock
DSI — Development Serial Input
DSO — Development Serial Output
EXTAL — External Crystal Oscillator Connection
FC[2:0] — Function Codes
$\overline{FREEZE}$ — Freeze
$\overline{HALT}$ — Halt
$\overline{IFETCH}$ — Instruction Fetch
$\overline{IPIPE}$ — Instruction Pipeline
$\overline{IRQ}$[7:1] — Interrupt Request
MISO — Master In Slave Out
MODCLK — Clock Mode Select
MOSI — Master Out Slave In
PC[6:0] — SIM I/O Port C
PCS[3:0] — Peripheral Chip Selects
PE[7:0] — SIM I/O Port E
PF[7:0] — SIM I/O Port F
PQS[7:0] — QSM I/O Port
QUOT — Quotient Out
R/$\overline{W}$ — Read/Write
$\overline{RESET}$ — Reset
$\overline{RMC}$ — Read-Modify-Write Cycle
RXD — SCI Receive Data
SCK — QSPI Serial Clock
SIZ[1:0] — Size
$\overline{SS}$ — Slave Select
T2CLK — TPU Clock In
TPUCH[15:0] — TPU Channel Signals
TSC — Three-State Control
TXD — SCI Transmit Data
XFC — External Filter Capacitor
XTAL — External Crystal Oscillator Connection

2.4 Register Mnemonics

CFSR[0:3] — Channel Function Select Registers [0:3]
CIER — Channel Interrupt Enable Register
CISR — Channel Interrupt Status Register
CPR[0:1] — Channel Priority Registers [0:1]
CREG — Test Control Register C
CR[0:F] — QSM Command RAM
CSBARBT — Chip-Select Base Address Register Boot ROM
CSBAR[0:10] — Chip-Select Base Address Registers [0:10]
CSORBT — Chip-Select Option Register Boot ROM
CSOR[0:10] — Chip-Select Option Registers [0:10]
CSPAR[0:1] — Chip-Select Pin Assignment Registers [0:1]
DCNR — Decoded Channel Number Register
DDRE — Port E Data Direction Register
DDRF — Port F Data Direction Register
DDRQS — Port QS Data Direction Register
DREG — SIM Test Module Distributed Register
DSCR — Development Support Control Register
DSSR — Development Support Status Register
HSQR[0:1] — Host Sequence Registers [0:1]
HSRR[0:1] — Host Service Request Registers [0:1]
LR — Link Register
PEPAR — Port E Pin Assignment Register
PFPAR — Port F Pin Assignment Register
PICR — Periodic Interrupt Control Register
PITR — Periodic Interrupt Timer Register
PORTC — Port C Data Register
PORTE — Port E Data Register
PORTF — Port F Data Register
PORTQS — Port QS Data Register
PQSPAR — Port QS Pin Assignment Register
QILR — QSM Interrupt Level Register
QIVR — QSM Interrupt Vector Register
QSMCR — QSM Configuration Register
QTEST — QSM Test Register
RR[0:F] — QSM Receive Data RAM
RSR — Reset Status Register
SCCR[0:1] — SCI Control Registers [0:1]
SCDR — SCI Data Register
SCSR — SCI Status Register
SGLR — Service Grant Latch Register
SIMCR — SIM Module Configuration Register
SIMTR — System Integration Test Register
SIMTRE — System Integration Test Register (ECLK)
SPCR[0:3] — QSPI Control Registers [0:3]
SPSR — QSPI Status Register

SWSR — Software Watchdog Service Register
SYNCR — Clock Synthesizer Control Register
SYPCR — System Protection Control Register
TCR — TPU Test Configuration Register
TICR — TPU Interrupt Configuration Register
TPUMCR — TPU Module Configuration Register
TRAMBAR — TPURAM Base Address/Status Register
TRAMMCR — TPURAM Module Configuration Register
TRAMTST — TPURAM Test Register
TR[0:F] — QSM Transmit Data RAM
TSTMSRA — Test Module Master Shift Register A
TSTMSRB — Test Module Master Shift Register B
TSTRC — Test Module Repetition Counter
TSTSC — Test Module Shift Count Register

2.5 Conventions

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

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

Set refers specifically to establishing logic level one on a bit or bits.

Clear refers specifically to establishing logic level zero on a bit or bits.

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

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

A specific mnemonic within a range is referred to by mnemonic and number. A15 is bit 15 of accumulator A; ADDR7 is line 7 of the address bus; CSOR0 is chip-select option register 0. **A range of mnemonics** is referred to by mnemonic and the numbers that define the range. AM[35:30] are bits 35 to 30 of accumulator M; CSOR[0:5] are the first six option registers

Parentheses are used to indicate the content of a register or memory location, rather than the register or memory location itself. (A) is the content of accumulator A. (M : M + 1) is the content of the word at address M.

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

LSW means least significant word or words. **MSW** means most significant word or words.

ADDR is the address bus. ADDR[7:0] are the eight LSB of the address bus.

DATA is the data bus. DATA[15:8] are the eight MSB of the data bus.

SECTION 3 OVERVIEW

This section contains information about the entire modular microcontroller. It lists the features of each module, shows device functional divisions and pin assignments, summarizes signal and pin functions, discusses the intermodule bus, and provides system memory maps. Timing and electrical specifications for the entire microcontroller and for individual modules are provided in **APPENDIX A ELECTRICAL CHARACTERISTICS**. Comprehensive module register descriptions and memory maps are provided in **APPENDIX D REGISTER SUMMARY**.

3.1 MC68332 Features

The following paragraphs highlight capabilities of each of the microcontroller modules. Each module is discussed separately in a subsequent section of this user's manual.

3.1.1 System Integration Module (SIM)

- External Bus Support
- Programmable Chip-Select Outputs
- System Protection Logic
- Watchdog Timer, Clock Monitor, and Bus Monitor
- System Protection Logic
- System Clock Based on 32.768-kHz Crystal for Low Power Operation
- Test/Debug Submodule for Factory/User Test and Development

3.1.2 Central Processing Unit (CPU)

- Upward Object Code Compatible
- New Instructions for Controller Applications
- 32-Bit Architecture
- Virtual Memory Implementation
- Loop Mode of Instruction Execution
- Table Lookup and Interpolate Instruction
- Improved Exception Handling for Controller Applications
- Trace on Change of Flow
- Hardware Breakpoint Signal, Background Mode
- Fully Static Operation

3.1.3 Time Processor Unit (TPU)

- Dedicated Microengine Operating Independently of CPU32
- 16 Independent, Programmable Channels and Pins
- Any Channel can Perform any Time Function
- Two Timer Count Registers with Programmable Prescalers
- Selectable Channel Priority Levels

3.1.4 Queued Serial Module (QSM)

- Enhanced Serial Communication Interface (SCI), Universal Asynchronous Receiver Transmitter (UART): Modulus Baud Rate, Parity
- Queued Serial Peripheral Interface (SPI): 80-Byte RAM, Up to 16 Automatic Transfers
- Dual Function I/O Ports
- Continuous Cycling, 8–16 Bits per Transfer

3.1.5 Static RAM Module with TPU Emulation Capability (TPURAM)

- 2-Kbytes of Static RAM
- May be Used as Normal RAM or TPU Microcode Emulation RAM

3.2 System Block Diagram and Pin Assignment Diagrams

Figure 3-1 is a functional diagram of the MCU. Although diagram blocks represent the relative size of the physical modules, there is not a one-to-one correspondence between location and size of blocks in the diagram and location and size of integrated-circuit modules. **Figure 3-2** shows the pin assignments of the 132-pin plastic surface-mount package. **Figure 3-3** shows the pin assignments of the 144-pin plastic surface-mount package. Refer to **APPENDIX B MECHANICAL DATA AND ORDERING INFORMATION** for package dimensions. All pin functions and signal names are shown in this drawing. Refer to subsequent paragraphs in this section for pin and signal descriptions.

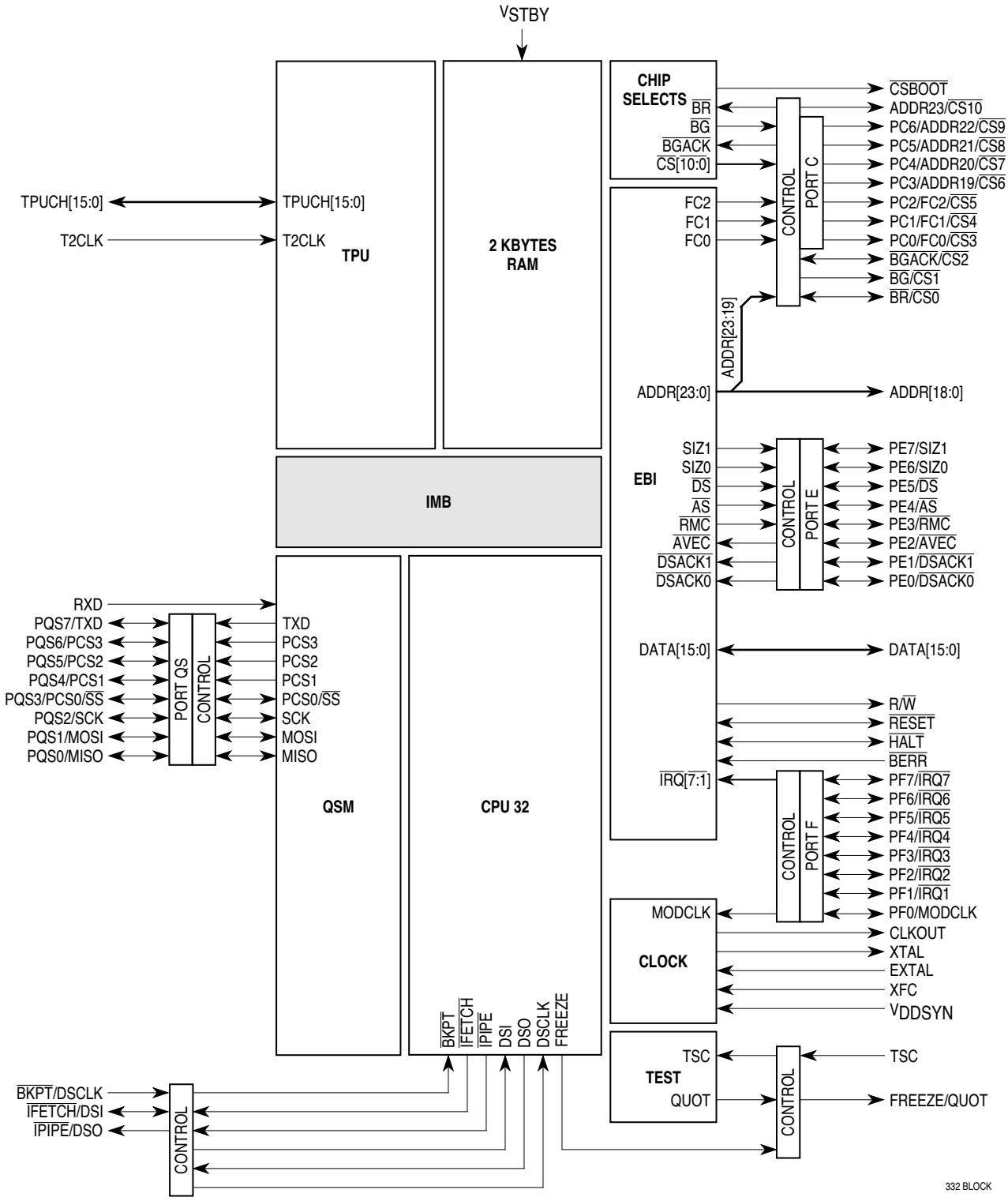


Figure 3-1 MCU Block Diagram

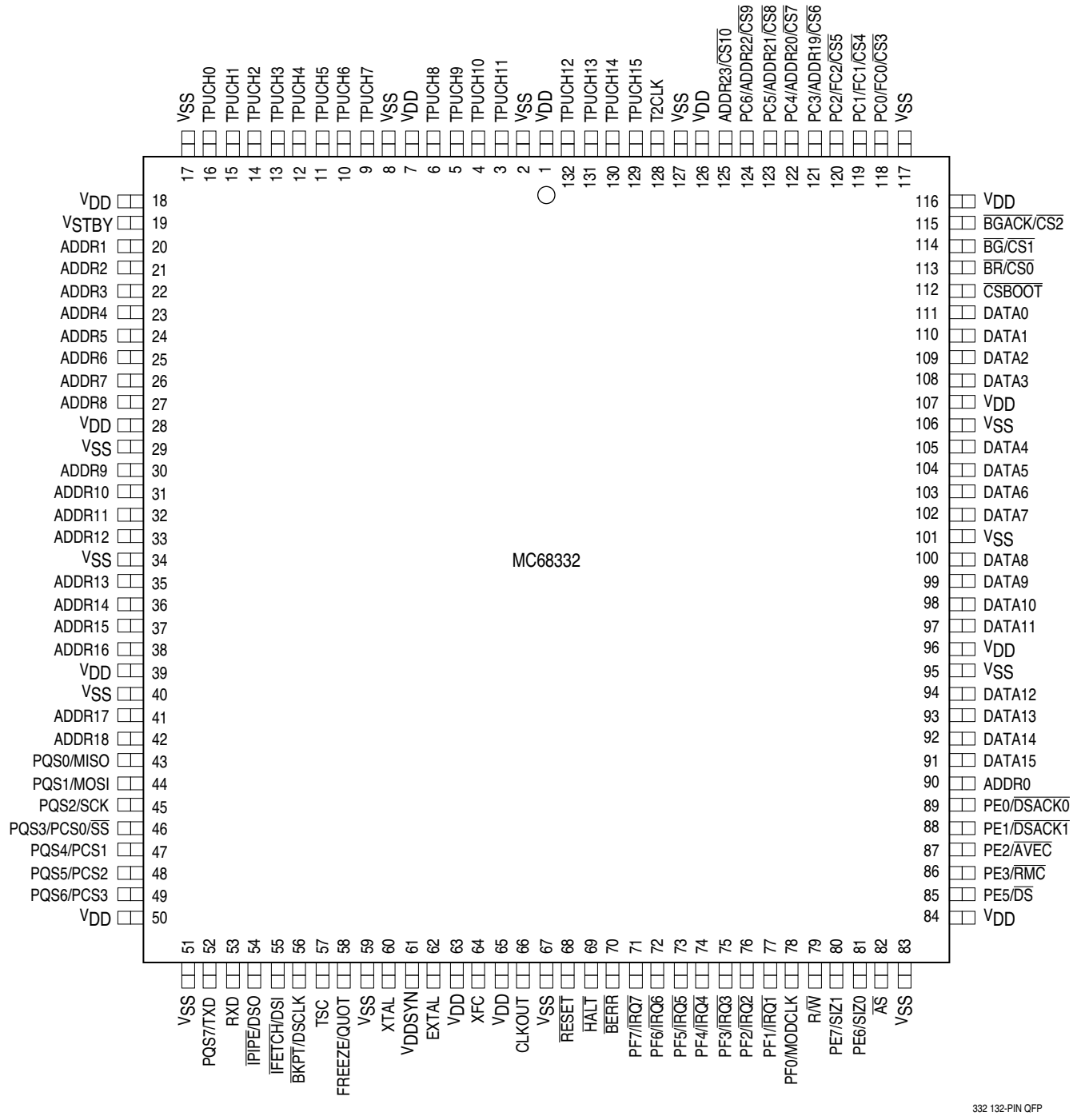


Figure 3-2 Pin Assignments for 132-Pin Package

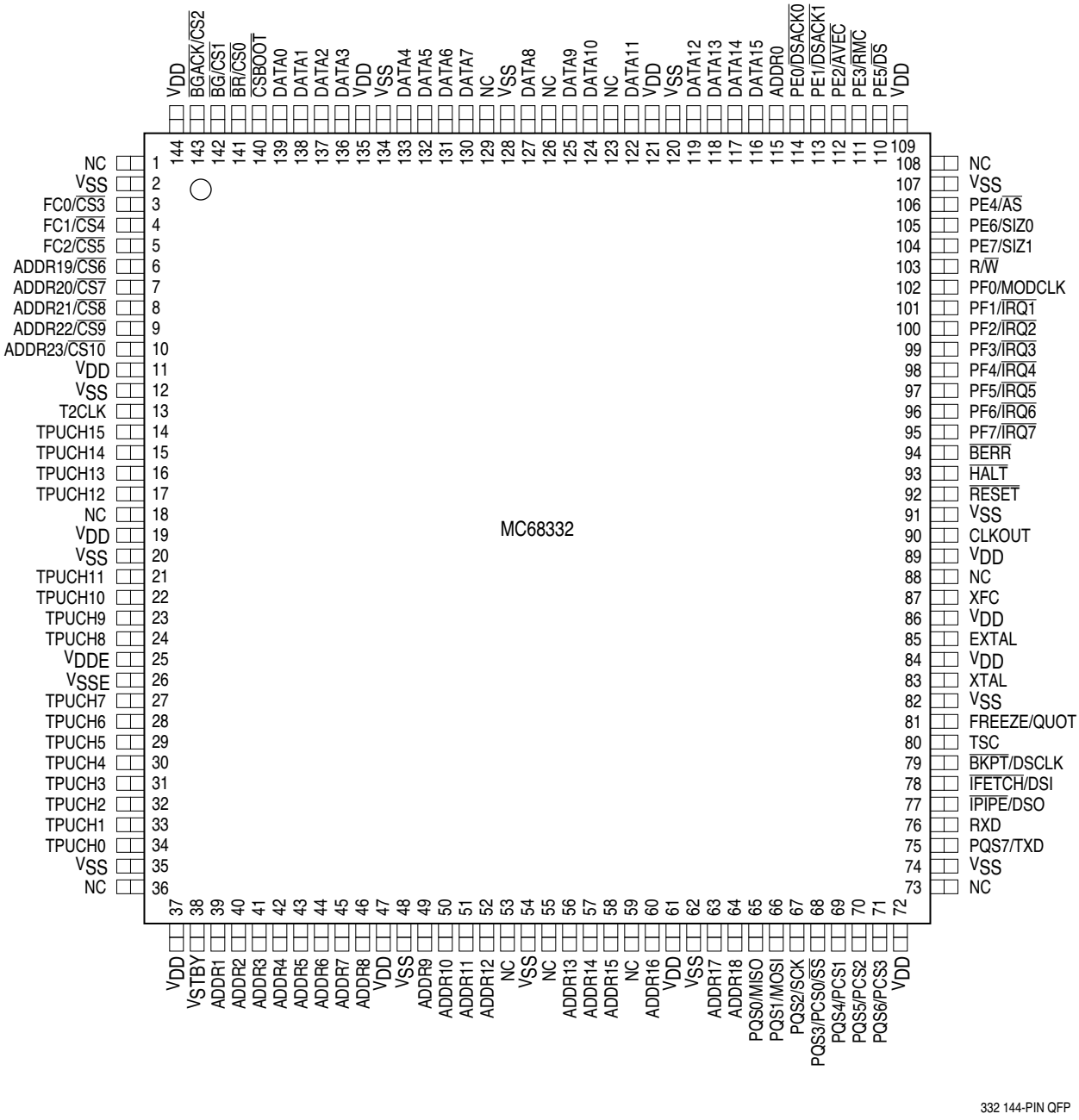


Figure 3-3 Pin Assignments for 144-Pin Package

3.3 Pin Descriptions

The following tables summarize functional characteristics of MCU pins. **Table 3-1** shows types of output drivers. **Table 3-2** shows all inputs and outputs. Digital inputs and outputs use CMOS logic levels. An entry in the Discrete I/O column indicates that a pin can also be used for general-purpose input, output, or both. The I/O port designation is given when it applies. **Table 3-3** shows characteristics of power pins. Refer to **Figure 3-1** for port organization.

Table 3-1 MCU Driver Types

Type	I/O	Description
A	O	Output-only signals that are always driven; no external pull-up required
Aw	O	Type A output with weak P-channel pull-up during reset
B	O	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.
Bo	O	Type B output that can be operated in an open-drain mode

Table 3-2 MCU Pin Characteristics

Pin Mnemonic	Output Driver	Input Synchronized	Input Hysteresis	Discrete I/O	Port Designation
ADDR23/$\overline{CS10}$/ECLK	A	Y	N	O	—
ADDR[22:19]/$\overline{CS}$[9:6]	A	Y	N	O	PC[6:3]
ADDR[18:0]	A	Y	N	—	—
$\overline{AS}$	B	Y	N	I/O	PE5
$\overline{AVEC}$	B	Y	N	I/O	PE2
$\overline{BERR}$	B	Y	N	—	—
$\overline{BG}$/$\overline{CS1}$	B	—	—	—	—
$\overline{BGACK}$/$\overline{CS2}$	B	Y	N	—	—
$\overline{BKPT}$/DSCLK	—	Y	Y	—	—
$\overline{BR}$/$\overline{CS0}$	B	Y	N	—	—
CLKOUT	A	—	—	—	—
$\overline{CSBOOT}$	B	—	—	—	—
DATA[15:0][1]	Aw	Y	N	—	—
$\overline{DS}$	B	Y	N	I/O	PE4
$\overline{DSACK1}$	B	Y	N	I/O	PE1
$\overline{DSACK0}$	B	Y	N	I/O	PE0
DSI/$\overline{IFETCH}$	A	Y	Y	—	—
DSO/$\overline{IPIPE}$	A	—	—	—	—
EXTAL[2]	—	—	Special	—	—
FC[2:0]/$\overline{CS}$[5:3]	A	Y	N	O	PC[2:0]
FREEZE/QUOT	A	—	—	—	—
$\overline{HALT}$	Bo	Y	N	—	—
$\overline{IRQ}$[7:1]	B	Y	Y	I/O	PF[7:1]
MISO	Bo	Y	Y	I/O	PQS0
MODCLK[1]	B	Y	N	I/O	PF0
MOSI	Bo	Y	Y	I/O	PQS1
PCS0/$\overline{SS}$	Bo	Y	Y	I/O	PQS3
PCS[3:1]	Bo	Y	Y	I/O	PQS[6:4]
R/$\overline{W}$	A	Y	N	—	—
$\overline{RESET}$	Bo	Y	Y	—	—
$\overline{RMC}$	B	Y	N	I/O	PE3
RXD	—	N	N	—	—
SCK	Bo	Y	Y	I/O	PQS2
SIZ[1:0]	B	Y	N	I/O	PE[7:6]
T2CLK	—	Y	Y	—	—
TPUCH[15:0]	A	Y	Y	—	—
TSC	—	Y	Y	—	—

Table 3-2 MCU Pin Characteristics (Continued)

Pin Mnemonic	Output Driver	Input Synchronized	Input Hysteresis	Discrete I/O	Port Designation
TXD	Bo	Y	Y	I/O	PQS7
XFC[2]	—	—	—	Special	—
XTAL[2]	—	—	—	Special	—

NOTES:
1. DATA[15:0] are synchronized during reset only. MODCLK is synchronized only when used as an input port pin.
2. EXTAL, XFC, and XTAL are clock reference connections.

Table 3-3 MCU Power Connections

Pin Mnemonic	Description
V_{STBY}	Standby RAM Power
V_{DDSYN}	Clock Synthesizer Power
V_{SSE}/V_{DDE}	External Periphery Power (Source and Drain)
V_{SSI}/V_{DDI}	Internal Module Power (Source and Drain)

3.4 Signal Descriptions

The following tables define MCU signals. **Table 3-4** shows signal origin, type, and active state. **Table 3-5** describes signal functions. Both tables are sorted alphabetically by mnemonic. MCU pins often have multiple functions. More than one description can apply to a pin.

Table 3-4 MCU Signal Characteristics

Signal Name	MCU Module	Signal Type	Active State
ADDR[23:0]	SIM	Bus	—
$\overline{AS}$	SIM	Output	0
$\overline{AVEC}$	SIM	Input	0
$\overline{BERR}$	SIM	Input	0
$\overline{BG}$	SIM	Output	0
$\overline{BGACK}$	SIM	Input	0
$\overline{BKPT}$	CPU32	Input	0
$\overline{BR}$	SIM	Input	0
CLKOUT	SIM	Output	—
$\overline{CS[10:0]}$	SIM	Output	0
$\overline{CSBOOT}$	SIM	Output	0
DATA[15:0]	SIM	Bus	—
$\overline{DS}$	SIM	Output	0
$\overline{DSACK[1:0]}$	SIM	Input	0
DSCLK	CPU32	Input	Serial Clock
DSI	CPU32	Input	(Serial Data)
DSO	CPU32	Output	(Serial Data)
EXTAL	SIM	Input	—
FC[2:0]	SIM	Output	—
FREEZE	SIM	Output	1
$\overline{HALT}$	SIM	Input/Output	0

Table 3-4 MCU Signal Characteristics (Continued)

Signal Name	MCU Module	Signal Type	Active State
$\overline{\text{IFETCH}}$	CPU32	Output	—
$\overline{\text{IPIPE}}$	CPU32	Output	—
$\overline{\text{IRQ}}$[7:1]	SIM	Input	0
MISO	QSM	Input/Output	—
MODCLK	SIM	Input	—
MOSI	QSM	Input/Output	—
PC[6:0]	SIM	Output	(Port)
PCS[3:0]	QSM	Input/Output	—
PE[7:0]	SIM	Input/Output	(Port)
PF[7:0]	SIM	Input/Output	(Port)
PQS[7:0]	QSM	Input/Output	(Port)
QUOT	SIM	Output	—
$\overline{\text{RESET}}$	SIM	Input/Output	0
$\overline{\text{RMC}}$	SIM	Output	0
R/$\overline{\text{W}}$	SIM	Output	1/0
RXD	QSM	Input	—
SCK	QSM	Input/Output	—
SIZ[1:0]	SIM	Output	—
$\overline{\text{SS}}$	QSM	Input	0
T2CLK	TPU	Input	—
TPUCH[15:0]	TPU	Input/Output	1
TSC	SIM	Input	—
TXD	QSM	Output	—
XFC	SIM	Input	—
XTAL	SIM	Output	—

Table 3-5 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	$\overline{\text{BERR}}$	Indicates that a bus error has occurred
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 start-up ROM
Data Bus	DATA[15:0]	16-bit data bus
Data Strobe	$\overline{\text{DS}}$	During a read cycle, indicates when it is possible for an external device to place data on the data bus. During a write cycle, indicates that valid data is on the data bus.
Data and Size Acknowledge	$\overline{\text{DSACK}}$[1:0]	Provide asynchronous data transfers and dynamic bus sizing
Development Serial In, Out, Clock	DSI, DSO, DSCLK	Serial I/O and clock for background debugging mode
Crystal Oscillator	EXTAL, XTAL	Connections for clock synthesizer circuit reference; a crystal or an external oscillator can be used

Table 3-5 MCU Signal Function (Continued)

Signal Name	Mnemonic	Function
Function Codes	FC[2:0]	Identify processor state and current address space
Freeze	FREEZE	Indicates that the CPU has entered background mode
Halt	$\overline{\text{HALT}}$	Suspend external bus activity
Instruction Pipeline	$\overline{\text{IPIPE}}$, $\overline{\text{IFETCH}}$	Indicate instruction pipeline activity
Interrupt Request Level	$\overline{\text{IRQ}}$[7:1]	Provides an interrupt priority level to 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 the source and type of system clock
Master Out Slave In	MOSI	Serial output from QSPI in master mode; serial input to QSPI in slave mode
Port C	PC[6:0]	SIM digital output port signals
Auxiliary Timer Clock Input	PCLK	External clock dedicated to the GPT
Peripheral Chip Select	PCS[3:0]	QSPI peripheral chip selects
Port E	PE[7:0]	SIM digital I/O port signals
Port F	PF[7:0]	SIM digital I/O 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	$\overline{\text{RESET}}$	System reset
Read-Modify-Write Cycle	$\overline{\text{RMC}}$	Indicates an indivisible read-modify-write instruction
Read/Write	R/$\overline{\text{W}}$	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	$\overline{\text{SS}}$	Causes serial transmission when QSPI is in slave mode; causes mode fault in master mode
TCR2 Clock	T2CLK	External clock source for TCR2 counter
TPU Channel Pins	TPUCH[15:0]	Bidirectional pins associated with TPU channels
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

3.5 Intermodule Bus

The intermodule bus (IMB) is a standardized bus developed to facilitate both design and operation of modular microcontrollers. It contains circuitry to support 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 through the IMB. The IMB in the MCU uses 24 address and 16 data lines.

3.6 System Memory Map

Figure 3-4 through **Figure 3-8** are MCU memory maps. **Figure 3-4** shows IMB addresses of internal registers. **Figure 3-5** through **Figure 3-8** show system memory maps that use different external decoding schemes.

3.6.1 Internal Register Map

In **Figure 3-4**, IMB ADDR[23:20] are represented by the letter Y. The value represented by Y determines the base address of MCU module control registers. In M68300 microcontrollers, Y is equal to M111, where M is the logic state of the module mapping (MM) bit in the system integration module configuration register (SIMCR).

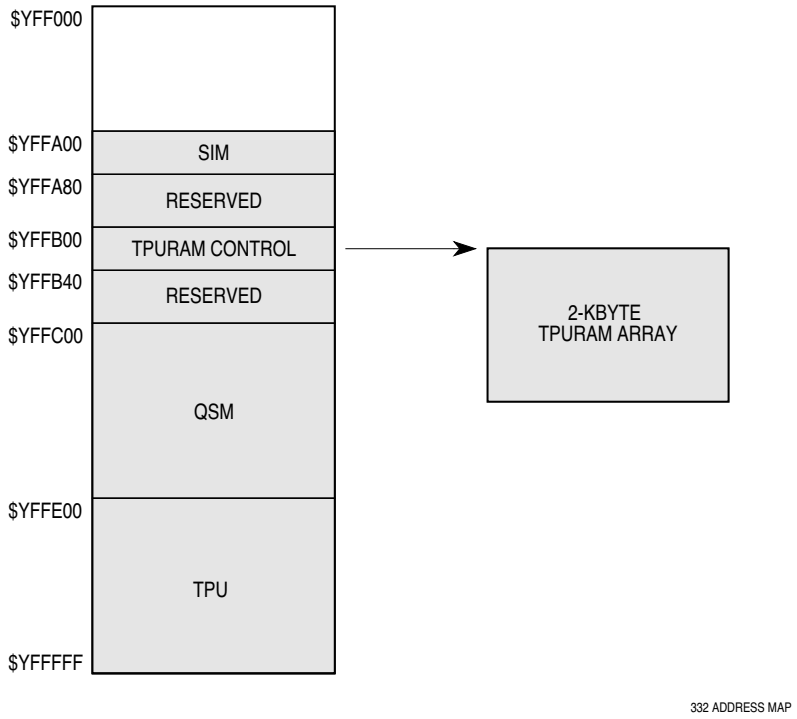
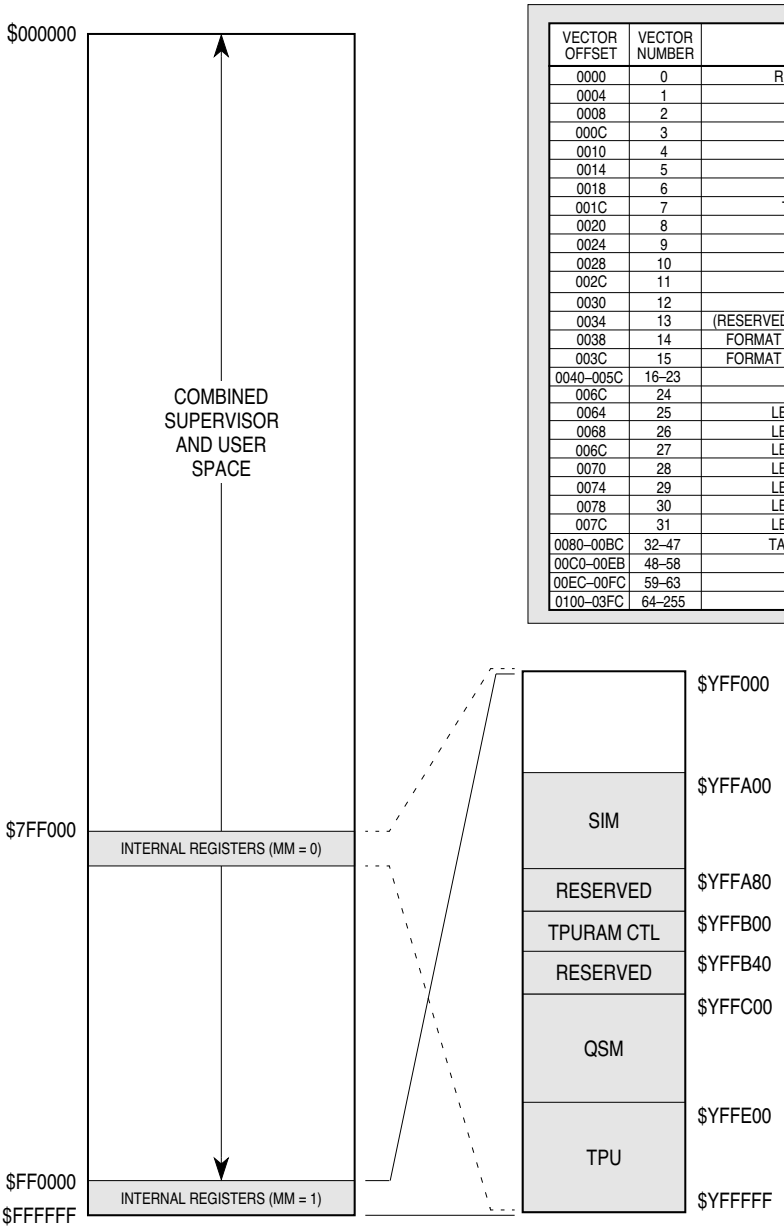


Figure 3-4 Internal Register Memory Map

3.6.2 Address Space Maps

Figure 3-5 shows a single memory space. Function codes FC[2:0] are not decoded externally so that separate user/supervisor or program/data spaces are not provided. In **Figure 3-6**, FC2 is decoded, resulting in separate supervisor and user spaces. FC[1:0] are not decoded, so that separate program and data spaces are not provided. In **Figure 3-7** and **Figure 3-8**, FC[2:0] are decoded, resulting in four separate memory spaces: supervisor/program, supervisor/data, user/program and user/data.

All exception vectors are located in supervisor data space, except the reset vector, which is located in supervisor program space. Only the initial reset vector is fixed in the processor's memory map. Once initialization is complete, there are no fixed assignments. Since the vector base register (VBR) provides the base address of the vector table, the vector table can be located anywhere in memory. Refer to **SECTION 5 CENTRAL PROCESSING UNIT** for more information concerning memory management, extended addressing, and exception processing. Refer to **SECTION 4 SYSTEM INTEGRATION MODULE** for more information concerning function codes and address space types.

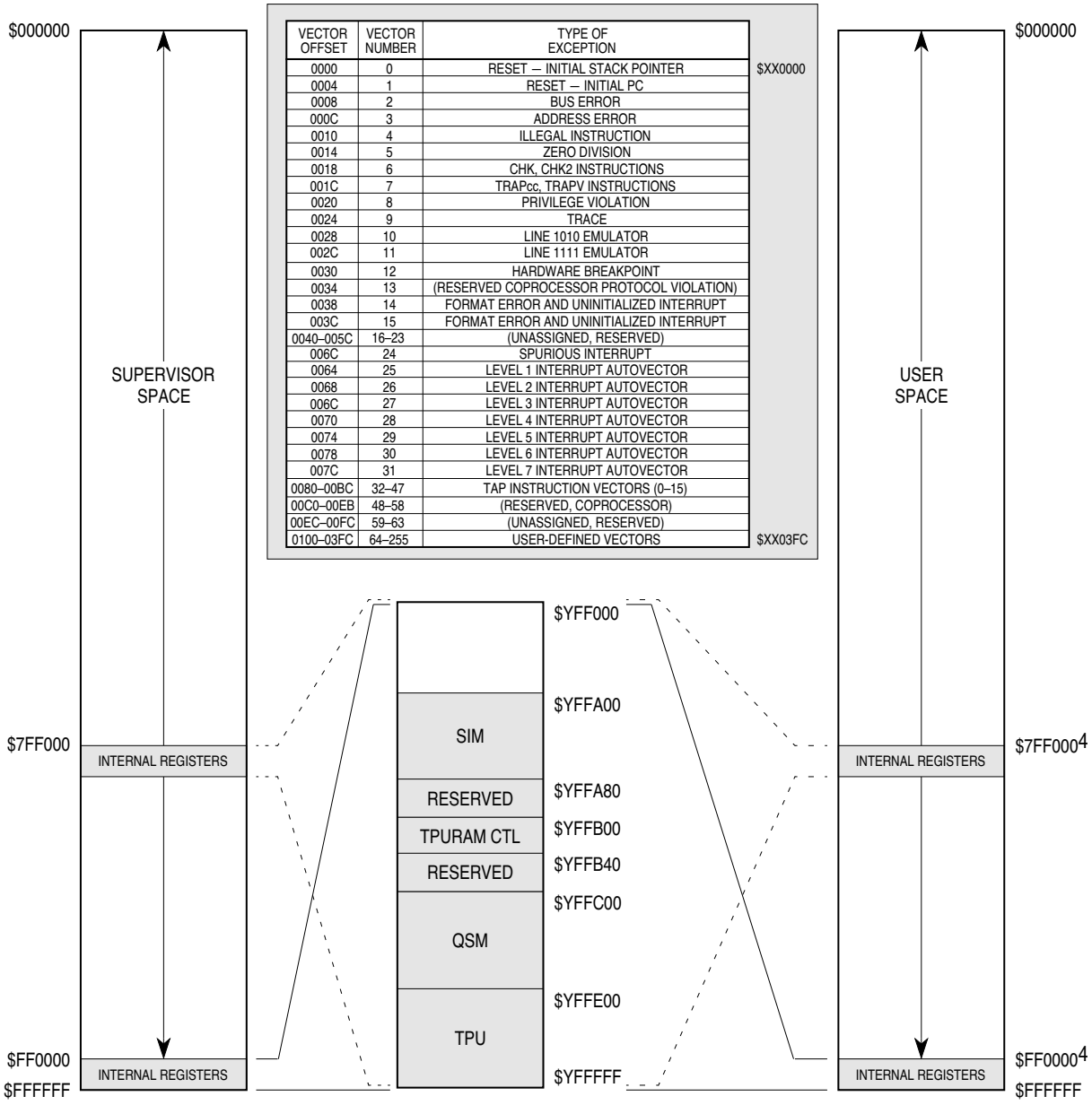


VECTOR OFFSET	VECTOR NUMBER	TYPE OF EXCEPTION	
0000	0	RESET — INITIAL STACK POINTER	$XX0000
0004	1	RESET — INITIAL PC	
0008	2	BUS ERROR	
000C	3	ADDRESS ERROR	
0010	4	ILLEGAL INSTRUCTION	
0014	5	ZERO DIVISION	
0018	6	CHK, CHK2 INSTRUCTIONS	
001C	7	TRAPcc, TRAPV INSTRUCTIONS	
0020	8	PRIVILEGE VIOLATION	
0024	9	TRACE	
0028	10	LINE 1010 EMULATOR	
002C	11	LINE 1111 EMULATOR	
0030	12	HARDWARE BREAKPOINT	
0034	13	(RESERVED COPROCESSOR PROTOCOL VIOLATION)	
0038	14	FORMAT ERROR AND UNINITIALIZED INTERRUPT	
003C	15	FORMAT ERROR AND UNINITIALIZED INTERRUPT	
0040–005C	16–23	(UNASSIGNED, RESERVED)	
006C	24	SPURIOUS INTERRUPT	
0064	25	LEVEL 1 INTERRUPT AUTOVECTOR	
0068	26	LEVEL 2 INTERRUPT AUTOVECTOR	
006C	27	LEVEL 3 INTERRUPT AUTOVECTOR	
0070	28	LEVEL 4 INTERRUPT AUTOVECTOR	
0074	29	LEVEL 5 INTERRUPT AUTOVECTOR	
0078	30	LEVEL 6 INTERRUPT AUTOVECTOR	
007C	31	LEVEL 7 INTERRUPT AUTOVECTOR	
0080–00BC	32–47	TAP INSTRUCTION VECTORS (0–15)	
00C0–00EB	48–58	(RESERVED, COPROCESSOR)	
00EC–00FC	59–63	(UNASSIGNED, RESERVED)	
0100–03FC	64–255	USER-DEFINED VECTORS	$XX03FC

NOTES:
1. Location of the exception vector table is determined by the vector base register. The vector address is the sum of the vector base register and the vector offset.
2. Location of the module control registers is determined by the state of the module mapping (MM) bit in the SIM configure register. Y = M111, where M is the state of the MM bit.
3. Unused addresses within the internal register block are mapped externally. "RESERVED" blocks are not mapped externally.

332 S/U COMB MAP

Figure 3-5 Overall Memory Map

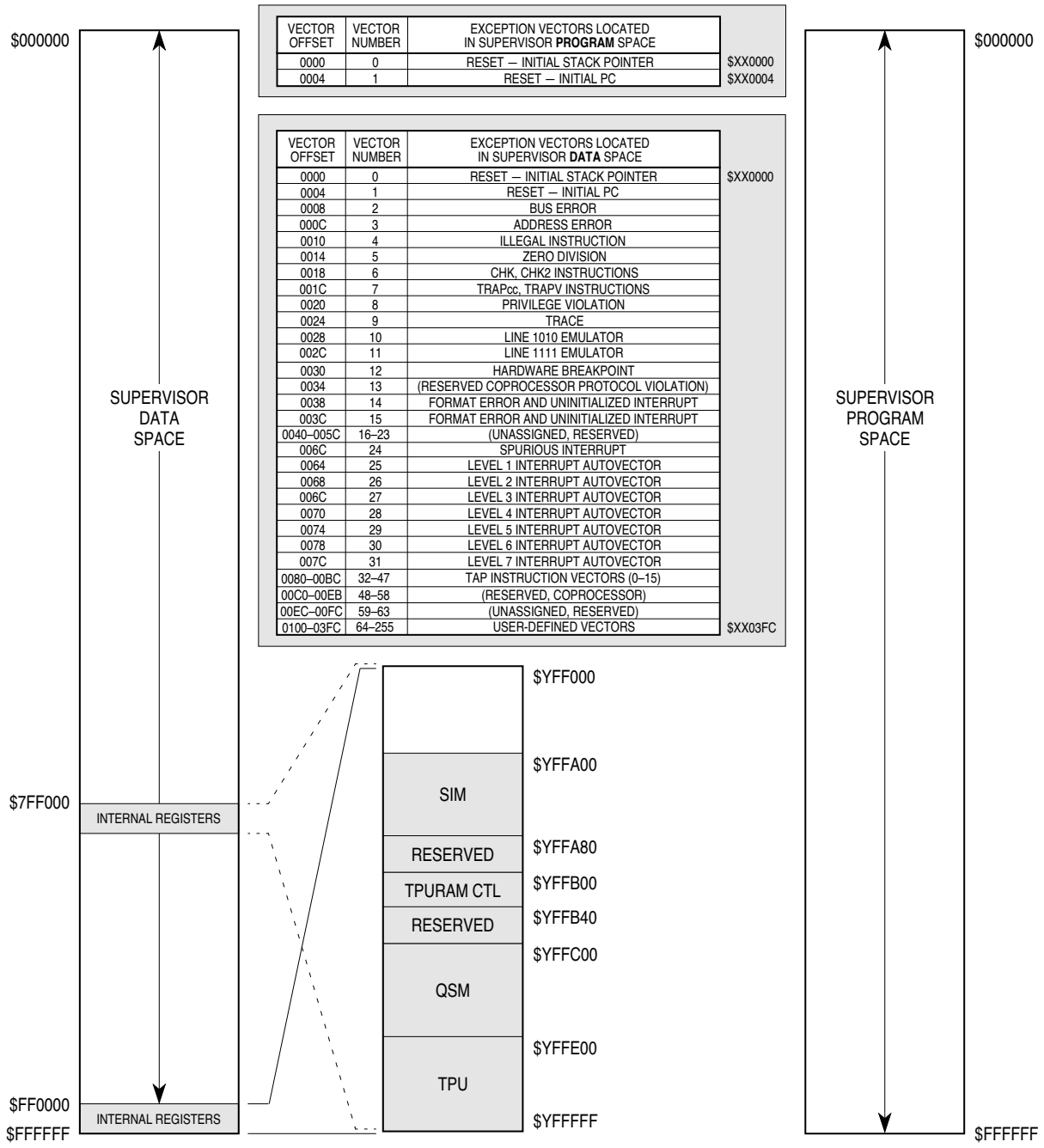


VECTOR OFFSET	VECTOR NUMBER	TYPE OF EXCEPTION	
0000	0	RESET — INITIAL STACK POINTER	$XX0000
0004	1	RESET — INITIAL PC	
0008	2	BUS ERROR	
000C	3	ADDRESS ERROR	
0010	4	ILLEGAL INSTRUCTION	
0014	5	ZERO DIVISION	
0018	6	CHK, CHK2 INSTRUCTIONS	
001C	7	TRAPcc, TRAPV INSTRUCTIONS	
0020	8	PRIVILEGE VIOLATION	
0024	9	TRACE	
0028	10	LINE 1010 EMULATOR	
002C	11	LINE 1111 EMULATOR	
0030	12	HARDWARE BREAKPOINT	
0034	13	(RESERVED COPROCESSOR PROTOCOL VIOLATION)	
0038	14	FORMAT ERROR AND UNINITIALIZED INTERRUPT	
003C	15	FORMAT ERROR AND UNINITIALIZED INTERRUPT	
0040–005C	16–23	(UNASSIGNED, RESERVED)	
006C	24	SPURIOUS INTERRUPT	
0064	25	LEVEL 1 INTERRUPT AUTOVECTOR	
0068	26	LEVEL 2 INTERRUPT AUTOVECTOR	
006C	27	LEVEL 3 INTERRUPT AUTOVECTOR	
0070	28	LEVEL 4 INTERRUPT AUTOVECTOR	
0074	29	LEVEL 5 INTERRUPT AUTOVECTOR	
0078	30	LEVEL 6 INTERRUPT AUTOVECTOR	
007C	31	LEVEL 7 INTERRUPT AUTOVECTOR	
0080–00BC	32–47	TAP INSTRUCTION VECTORS (0–15)	
00C0–00EB	48–58	(RESERVED, COPROCESSOR)	
00EC–00FC	59–63	(UNASSIGNED, RESERVED)	
0100–03FC	64–255	USER-DEFINED VECTORS	$XX03FC

NOTES:
1. Location of the exception vector table is determined by the vector base register. The vector address is the sum of the vector base register and the vector offset.
2. Location of the module control registers is determined by the state of the module mapping (MM) bit in the SIM configure register. Y = M111, where M is the state of the MM bit.
3. Unused addresses within the internal register block are mapped externally. "RESERVED" blocks are not mapped externally.
4. Some internal registers are not available in user space.

332 S/U SEP MAP

Figure 3-6 Separate Supervisor and User Space Map



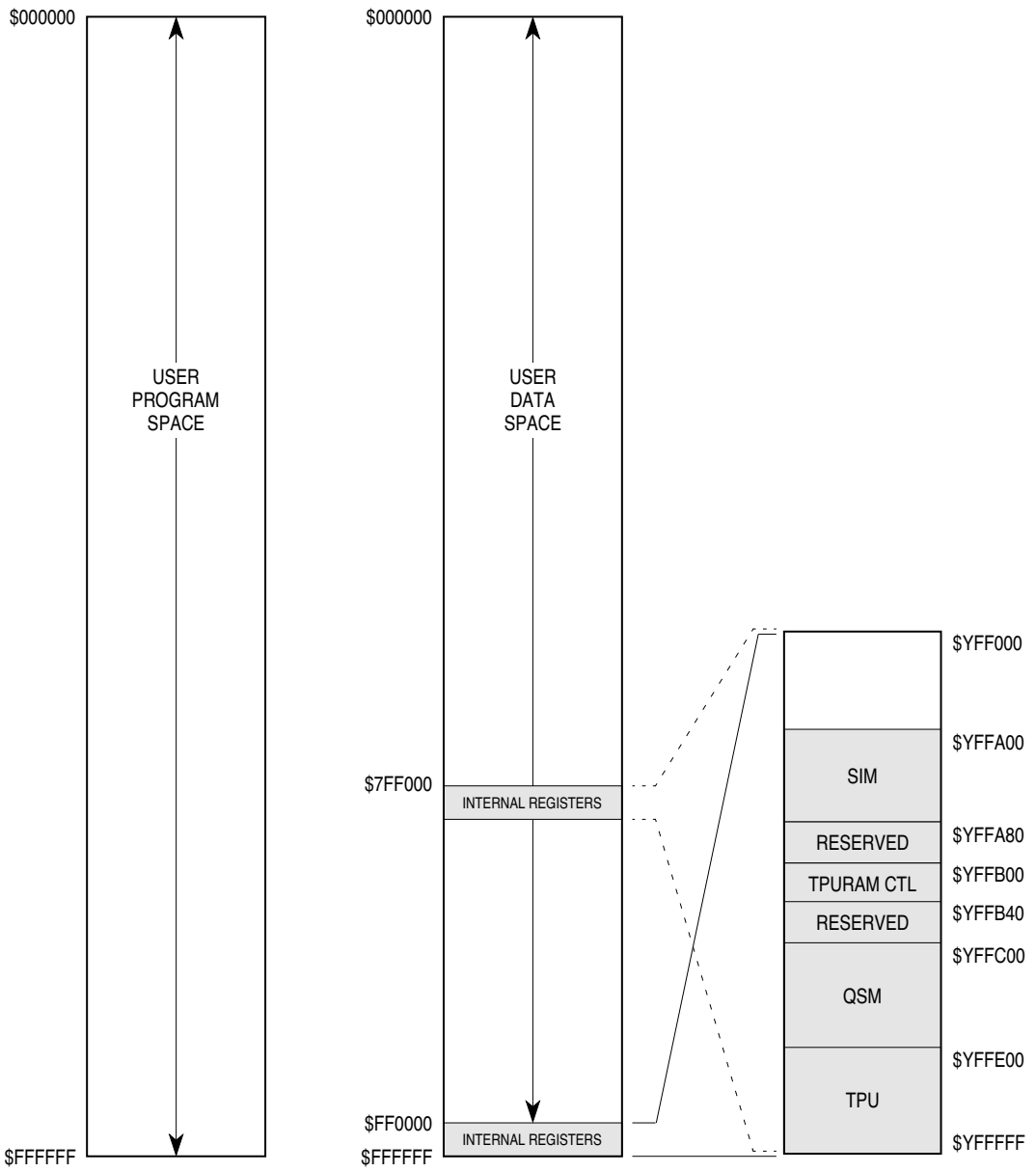
VECTOR OFFSET	VECTOR NUMBER	EXCEPTION VECTORS LOCATED IN SUPERVISOR **PROGRAM** SPACE	
0000	0	RESET — INITIAL STACK POINTER	$XX0000
0004	1	RESET — INITIAL PC	$XX0004

VECTOR OFFSET	VECTOR NUMBER	EXCEPTION VECTORS LOCATED IN SUPERVISOR **DATA** SPACE	
0000	0	RESET — INITIAL STACK POINTER	$XX0000
0004	1	RESET — INITIAL PC	
0008	2	BUS ERROR	
000C	3	ADDRESS ERROR	
0010	4	ILLEGAL INSTRUCTION	
0014	5	ZERO DIVISION	
0018	6	CHK, CHK2 INSTRUCTIONS	
001C	7	TRAPcc, TRAPV INSTRUCTIONS	
0020	8	PRIVILEGE VIOLATION	
0024	9	TRACE	
0028	10	LINE 1010 EMULATOR	
002C	11	LINE 1111 EMULATOR	
0030	12	HARDWARE BREAKPOINT	
0034	13	(RESERVED COPROCESSOR PROTOCOL VIOLATION)	
0038	14	FORMAT ERROR AND UNINITIALIZED INTERRUPT	
003C	15	FORMAT ERROR AND UNINITIALIZED INTERRUPT	
0040–005C	16–23	(UNASSIGNED, RESERVED)	
006C	24	SPURIOUS INTERRUPT	
0064	25	LEVEL 1 INTERRUPT AUTOVECTOR	
0068	26	LEVEL 2 INTERRUPT AUTOVECTOR	
006C	27	LEVEL 3 INTERRUPT AUTOVECTOR	
0070	28	LEVEL 4 INTERRUPT AUTOVECTOR	
0074	29	LEVEL 5 INTERRUPT AUTOVECTOR	
0078	30	LEVEL 6 INTERRUPT AUTOVECTOR	
007C	31	LEVEL 7 INTERRUPT AUTOVECTOR	
0080–00BC	32–47	TAP INSTRUCTION VECTORS (0–15)	
00C0–00EB	48–58	(RESERVED, COPROCESSOR)	
00EC–00FC	59–63	(UNASSIGNED, RESERVED)	
0100–03FC	64–255	USER-DEFINED VECTORS	$XX03FC

NOTES:
1. Location of the exception vector table is determined by the vector base register. The vector address is the sum of the vector base register and the vector offset.
2. Location of the module control registers is determined by the state of the module mapping (MM) bit in the SIM configure register. Y = M111, where M is the state of the MM bit.
3. Unused addresses within the internal register block are mapped externally. "RESERVED" blocks are not mapped externally.
4. Some internal registers are not available in user space.

332 SUPER P/D MAP

Figure 3-7 Supervisor Space (Separate Program/Data Space) Map



NOTES:
1. Location of the exception vector table is determined by the vector base register. The vector address is the sum of the vector base register and the vector offset.
2. Unused addresses within the internal register block are mapped externally. "RESERVED" blocks are not mapped externally.
3. Some internal registers are not available in user space.

Figure 3-8 User Space (Separate Program/Data Space) Map

3.7 System Reset

The following information is a concise reference only. MC68332 system reset is a complex operation. To understand operation during and after reset, refer to **SECTION 4 SYSTEM INTEGRATION MODULE**, paragraph **4.6 Reset** for more complete discussion of the reset function.

3.7.1 SIM Reset Mode Selection

The logic states of certain data bus pins during reset determine SIM operating configuration. In addition, the state of the MODCLK pin determines system clock source and the state of the $\overline{\text{BKPT}}$ pin determines what happens during subsequent breakpoint assertions. **Table 3-6** is a summary of reset mode selection options.

Table 3-6 SIM 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	$\overline{\text{DSACK0}}$, $\overline{\text{DSACK1}}$, $\overline{\text{AVEC}}$, $\overline{\text{DS}}$, $\overline{\text{AS}}$, $\overline{\text{SIZ}}$[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

3.7.2 MCU Module Pin Function During Reset

Generally, pins associated with modules other than the SIM 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. **Table 3-7** is a summary of module pin function out of reset.

Table 3-7 Module Pin Functions

Module	Pin Mnemonic	Function
CPU32	DSI/$\overline{\text{IFETCH}}$	DSI/$\overline{\text{IFETCH}}$
	DSO/$\overline{\text{IPIPE}}$	DSO/$\overline{\text{IPIPE}}$
	$\overline{\text{BKPT}}$/DSCLK	$\overline{\text{BKPT}}$/DSCLK
TPU	TPUCH[15:0]	TPU Input
	T2CLK	TCR2 Clock
QSM	PQS7/TXD	Discrete Input
	PQS[6:4]/PCS[3:1]	Discrete Input
	PQS3/PCS0/$\overline{\text{SS}}$	Discrete Input
	PQS2/SCK	Discrete Input
	PQS1/MOSI	Discrete Input
	PQS0/MISO	Discrete Input
	RXD	RXD

SECTION 4 SYSTEM INTEGRATION MODULE

This section is an overview of SIM function. Refer to the *SIM Reference Manual* (SIMRM/AD) for a comprehensive discussion of SIM capabilities. Refer to **APPENDIX D REGISTER SUMMARY** for information concerning the SIM address map and register structure.

4.1 General

The system integration module (SIM) consists of five functional blocks. **Figure 4-1** is a block diagram of the SIM.

The system configuration and protection block controls configuration parameters and provides bus and software watchdog monitors. In addition, it provides a periodic interrupt generator to support execution of time-critical control routines.

The system clock generates clock signals used by the SIM, other IMB modules, and external devices.

The external bus interface handles the transfer of information between IMB modules and external address space. EBI pins can also be configured for use as general-purpose I/O ports E and F.

The chip-select block provides 12 chip-select signals. Each chip-select signal has an associated base register and option register that contain the programmable characteristics of that chip select. Chip-select pins can also be configured for use as general-purpose output port C.

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.

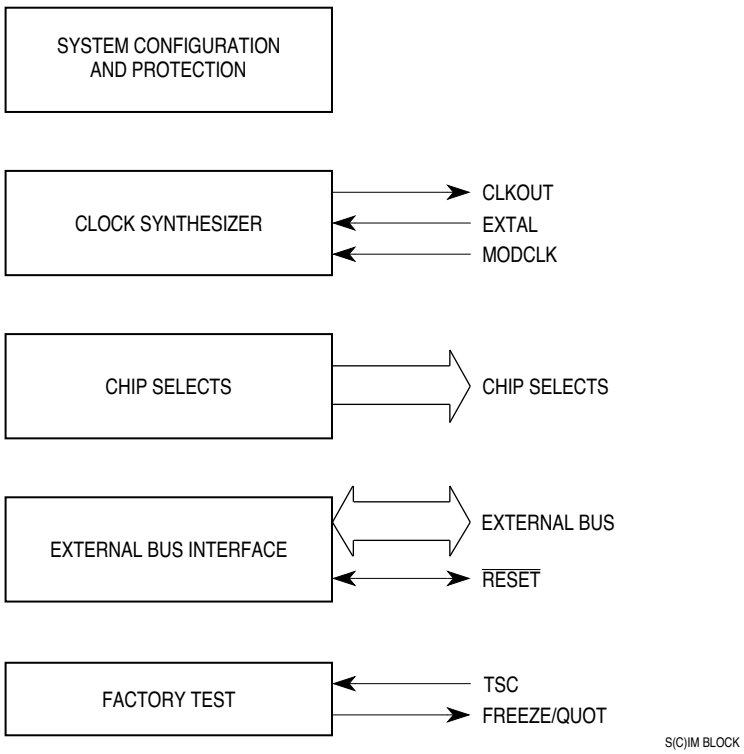


Figure 4-1 System Integration Module Block Diagram

4.2 System Configuration and Protection

The system configuration and protection functional block controls module configuration, preserves reset status, monitors internal activity, and provides periodic interrupt generation. **Figure 4-2** is a block diagram of the submodule.

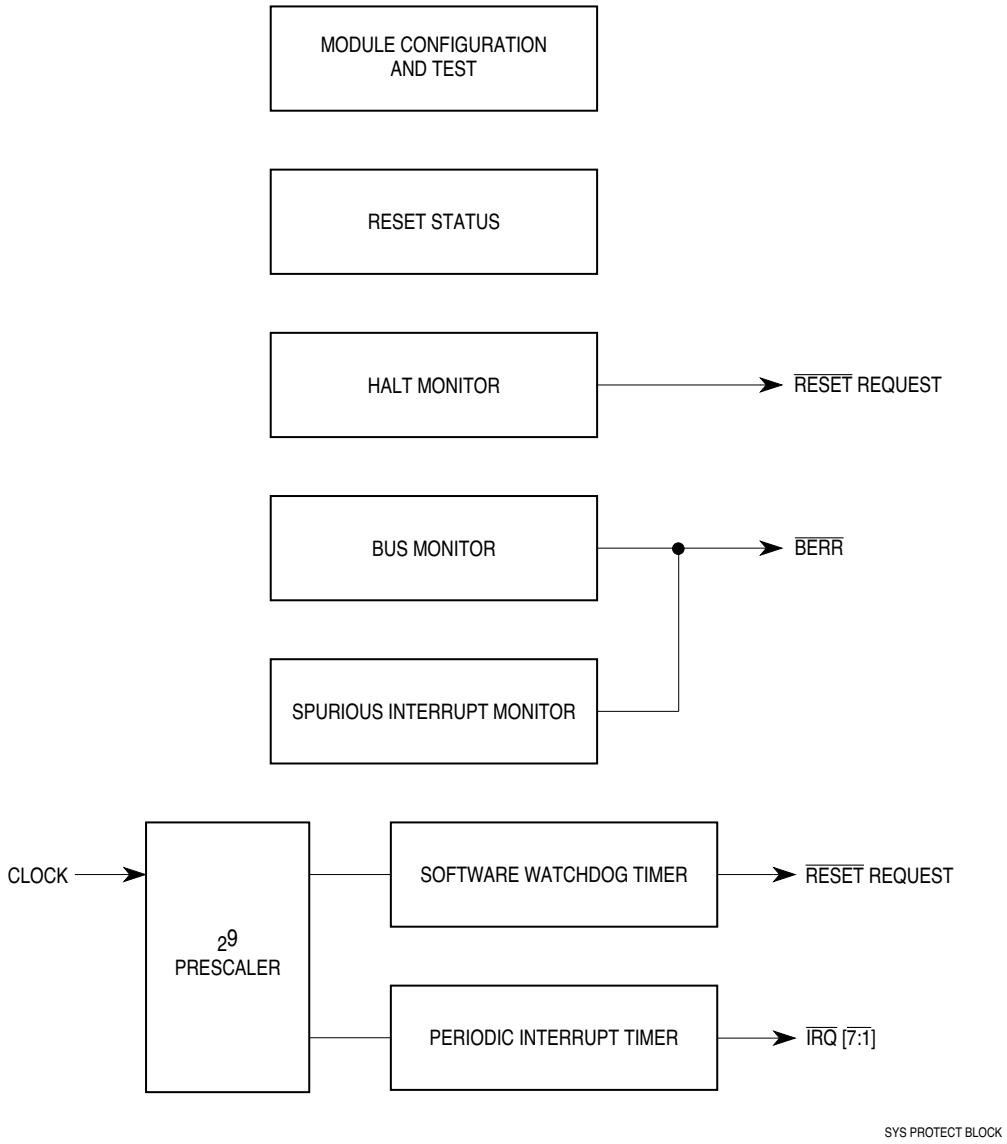


Figure 4-2 System Configuration and Protection

4.2.1 Module Mapping

Control registers for all the modules in the microcontroller are mapped into a 4-Kbyte block. The state of the module mapping bit (MM) in the SIM configuration register (SIMCR) determines where the control register block is located in the system memory map. When MM = 0, register addresses range from $7FF000 to $7FFFFF; when MM = 1, register addresses range from $FFF000 to $FFFFFF.

4.2.2 Interrupt Arbitration

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 in-

terrupt request is acknowledged, even when there is only a single request pending. For an interrupt to be serviced, the appropriate IARB field must have a non-zero value. If an interrupt request from a module with an IARB field value of %0000 is recognized, the CPU32 processes a spurious interrupt exception.

Because the SIM routes external interrupt requests to the CPU32, the SIM IARB field value is used for arbitration between internal and external interrupts of the same priority. The reset value of IARB for the SIM is %1111, and the reset IARB value for all other modules is %0000, which prevents SIM interrupts from being discarded during initialization. Refer to **4.7 Interrupts** for a discussion of interrupt arbitration.

4.2.3 Show Internal Cycles

A show cycle allows internal bus transfers to be monitored externally. The SHEN field in the SIMCR determines what the external bus interface does during internal transfer operations. **Table 4-1** shows whether data is driven externally, and whether external bus arbitration can occur. Refer to **4.5.6.2 Show Cycles** for more information.

Table 4-1 Show Cycle Enable Bits

SHEN	Action
00	Show cycles disabled, external arbitration enabled
01	Show cycles enabled, external arbitration disabled
10	Show cycles enabled, external arbitration enabled
11	Show cycles enabled, external arbitration enabled; internal activity is halted by a bus grant

4.2.4 Factory Test Mode

The internal IMB can serve as slave to an external master for direct module testing. This test mode is reserved for factory test. Slave mode is enabled by holding DATA11 low during reset. The slave enabled (SLVEN) bit is a read-only bit that shows the reset state of DATA11.

4.2.5 Register Access

The CPU32 can operate at either of two privilege levels. Supervisor level is more privileged than user level — all instructions and system resources are available at supervisor level, but access is restricted at user level. Effective use of privilege level can protect system resources from uncontrolled access. The state of the S bit in the CPU status register determines access level, and whether the user or supervisor stack pointer is used for stacking operations. The SUPV bit places SIM global registers in either supervisor or user data space. When SUPV = 0, registers with controlled access are accessible from either the user or supervisor privilege level; when SUPV = 1, registers with controlled access are restricted to supervisor access only.

4.2.6 Reset Status

The reset status register (RSR) latches internal MCU status during reset. Refer to **4.6.9 Reset Status Register** for more information.

4.2.7 Bus Monitor

The internal bus monitor checks data and size acknowledge ($\overline{\text{DSACK}}$) or autovector ($\overline{\text{AVEC}}$) signal response times during normal bus cycles. The monitor asserts the internal bus error ($\overline{\text{BERR}}$) signal when the response time is excessively long.

$\overline{\text{DSACK}}$ and $\overline{\text{AVEC}}$ response times are measured in clock cycles. Maximum allowable response time can be selected by setting the bus monitor timing (BMT) field in the system protection control register (SYPCR). **Table 4-2** shows the periods allowed.

Table 4-2 Bus Monitor Period

BMT	Bus Monitor Time-out Period
00	64 System Clocks
01	32 System Clocks
10	16 System Clocks
11	8 System Clocks

The monitor does not check $\overline{\text{DSACK}}$ response on the external bus unless the CPU32 initiates a 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.

When monitoring transfers to an 8-bit port, the bus monitor does not reset until both byte accesses of a word transfer are completed. Monitor time-out period must be at least twice the number of clocks that a single byte access requires.

4.2.8 Halt Monitor

The halt monitor responds to an assertion of the $\overline{\text{HALT}}$ signal on the internal bus. Refer to **4.5.5.2 Double Bus Faults** for more information. Halt monitor reset can be inhibited by the halt monitor (HME) bit in SYPCR.

4.2.9 Spurious Interrupt Monitor

During interrupt exception processing, the CPU32 normally acknowledges an interrupt request, recognizes the highest priority source, and then acquires a vector or responds to a request for autovectoring. The spurious interrupt monitor asserts the internal bus error signal ($\overline{\text{BERR}}$) if no interrupt arbitration occurs during interrupt exception processing. The assertion of $\overline{\text{BERR}}$ causes the CPU32 to load the spurious interrupt exception vector into the program counter. The spurious interrupt monitor cannot be disabled. Refer to **4.7 Interrupts** for further information. For detailed information about interrupt exception processing, refer to **SECTION 5 CENTRAL PROCESSING UNIT**.

4.2.10 Software Watchdog

The software watchdog is controlled by the software watchdog enable (SWE) bit in SYPCR. When enabled, the watchdog requires that a service sequence be written to software service register SWSR on a periodic basis. If servicing does not take place, the watchdog times out and asserts the reset signal.

Perform a software watchdog service sequence as follows:

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.

Watchdog clock rate is affected by the software watchdog prescale (SWP) and software watchdog timing (SWT) fields in SYPCR.

SWP determines system clock prescaling for the watchdog timer and determines that one of two options, either no prescaling or prescaling by a factor of 512, can be selected. The value of SWP is affected by the state of the MODCLK pin during reset, as shown in **Table 4-3**. System software can change SWP value.

Table 4-3 MODCLK Pin and SWP Bit During Reset

MODCLK	SWP
0 (External Clock)	1 (÷ 512)
1 (Internal Clock)	0 (÷ 1)

The SWT field selects the divide ratio used to establish software watchdog time-out period. Time-out period is given by the following equations.

$$\text{Time-out Period} = \frac{1}{\text{EXTAL Frequency} / \text{Divide Ratio}}$$

or

$$\text{Time-out Period} = \frac{\text{Divide Ratio}}{\text{EXTAL Frequency}}$$

Table 4-4 shows the ratio for each combination of SWP and SWT bits. When SWT[1:0] are modified, a watchdog service sequence must be performed before the new time-out period can take effect.

Table 4-4 Software Watchdog Ratio

SWP	SWT	Ratio
0	00	2^9
0	01	2^{11}
0	10	2^{13}
0	11	2^{15}
1	00	2^{18}
1	01	2^{20}
1	10	2^{22}
1	11	2^{24}

Figure 4-3 is a block diagram of the watchdog timer and the clock control for the periodic interrupt timer.

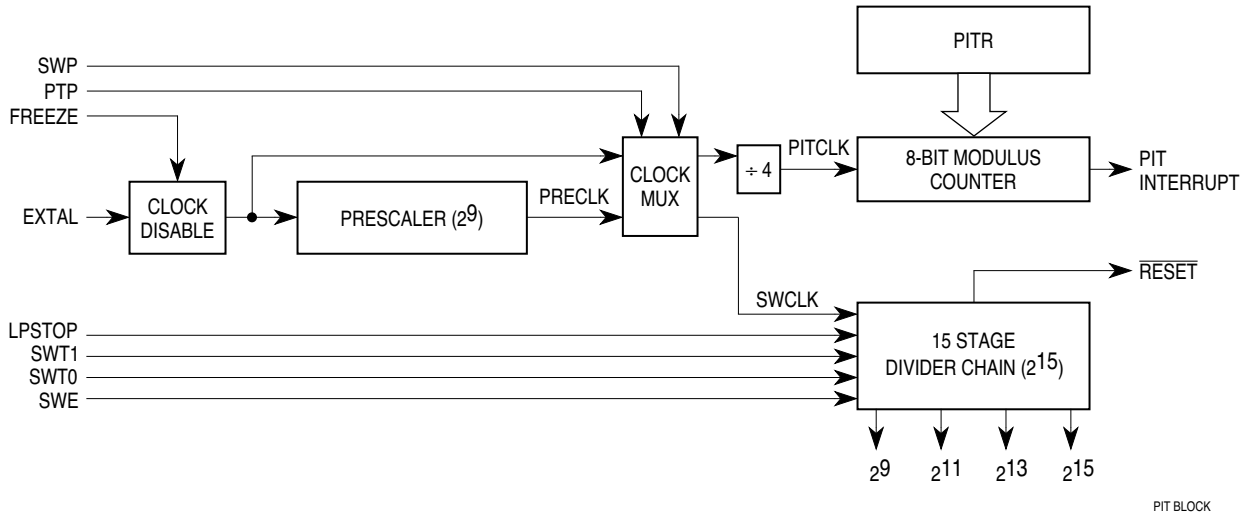


Figure 4-3 Periodic Interrupt Timer and Software Watchdog Timer

4.2.11 Periodic Interrupt Timer

The periodic interrupt timer allows the generation of interrupts of specific priority at predetermined intervals. This capability is often used to schedule control system tasks that must be performed within time constraints. The timer consists of a prescaler, a modulus counter, and registers that determine interrupt timing, priority and vector assignment. Refer to **SECTION 5 CENTRAL PROCESSING UNIT** for further information about interrupt exception processing.

The periodic interrupt modulus counter is clocked by a signal derived from the buffered crystal oscillator (EXTAL) input pin unless an external frequency source is used. The value of the periodic timer prescaler (PTP) bit in the periodic interrupt timer register (PITR) determines system clock prescaling for the watchdog timer. One of two options, either no prescaling, or prescaling by a factor of 512, can be selected. The value of PTP is affected by the state of the MODCLK pin during reset, as shown in **Table 4-5**. System software can change PTP value.

Table 4-5 MODCLK Pin and PTP Bit at Reset

MODCLK	PTP
0 (External Clock)	1 (÷ 512)
1 (Internal Clock)	0 (÷ 1)

Either clock signal (EXTAL or EXTAL ÷ 512) is divided by four before driving the modulus counter (PITCLK). The modulus counter is initialized by writing a value to the periodic timer modulus (PITM) field in the PITR. A zero value turns off the periodic timer. When the modulus counter value reaches zero, an interrupt is generated. The modulus counter is then reloaded with the value in PITM and counting repeats. If a new value is written to PITR, it is loaded into the modulus counter when the current count is completed.

Use the following expression to calculate timer period.

$$\text{PIT Period} = \frac{(\text{PIT Modulus})(\text{Prescaler Value})(4)}{\text{EXTAL Frequency}}$$

Interrupt priority and vectoring are determined by the values of the periodic interrupt request level (PIRQL) and periodic interrupt vector (PIV) fields in the periodic interrupt control register (PICR).

Content of PIRQL is compared to the CPU32 interrupt priority mask to determine whether the interrupt is recognized. **Table 4-6** shows priority of PIRQL values. Because of SIM hardware prioritization, a PIT interrupt is serviced before an external interrupt request of the same priority. The periodic timer continues to run when the interrupt is disabled.

Table 4-6 Periodic Interrupt Priority

PIRQL	Priority Level
000	Periodic Interrupt Disabled
001	Interrupt Priority Level 1
010	Interrupt Priority Level 2
011	Interrupt Priority Level 3
100	Interrupt Priority Level 4
101	Interrupt Priority Level 5
110	Interrupt Priority Level 6
111	Interrupt Priority Level 7

The PIV field contains the periodic interrupt vector. The vector is placed on the IMB when an interrupt request is made. The vector number used to calculate the address of the appropriate exception vector in the exception vector table. Reset value of the PIV field is $0F, which corresponds to the uninitialized interrupt exception vector.

4.2.12 Low-Power Stop Operation

When the CPU32 executes the LPSTOP instruction, the current interrupt priority mask is stored in the clock control logic, internal clocks are disabled according to the state of the STSIM bit in the SIMCR, and the MCU enters low-power stop mode. The bus monitor, halt monitor, and spurious interrupt monitor are all inactive during low-power stop.

During low-power stop, the clock input to the software watchdog timer is disabled and the timer stops. The software watchdog begins to run again on the first rising clock edge after low-power stop ends. The watchdog is not reset by low-power stop. A service sequence must be performed to reset the timer.

The periodic interrupt timer does not respond to the LPSTOP instruction, but continues to run during LPSTOP. To stop the periodic interrupt timer, PITR must be loaded with a zero value before the LPSTOP instruction is executed. A PIT interrupt, or an external interrupt request, can bring the MCU out of the low-power stop condition if it has a higher priority than the interrupt mask value stored in the clock control logic when low-power stop is initiated. LPSTOP can be terminated by a reset.

4.2.13 Freeze Operation

The FREEZE signal halts MCU operations during debugging. FREEZE is asserted internally by the CPU32 if a breakpoint occurs while background mode is enabled. When FREEZE is asserted, only the bus monitor, software watchdog, and periodic interrupt timer are affected. The halt monitor and spurious interrupt monitor continue to operate normally. Setting the freeze bus monitor (FRZBM) bit in the SIMCR disables the bus monitor when FREEZE is asserted, and setting the freeze software watchdog (FRZSW) bit disables the software watchdog and the periodic interrupt timer when FREEZE is asserted. When FRZSW is set, FREEZE assertion must be at least two times the PIT clock source period to ensure an accurate number of PIT counts.

4.3 System Clock

The system clock in the SIM 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 three ways. An internal phase-locked loop can synthesize the clock from either an internal reference or an external reference, or the clock signal can be input from an external frequency source. Keep these clock sources in mind while reading the rest of this section. **Figure 4-4** is a block diagram of the system clock. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for clock specifications.

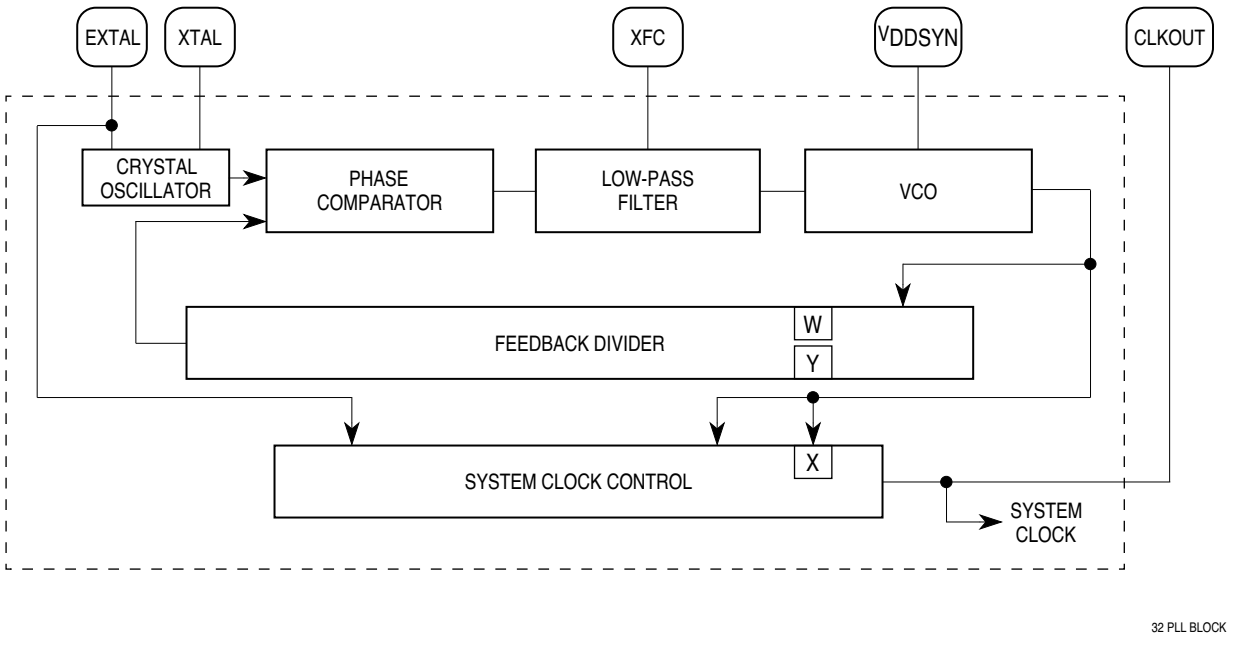
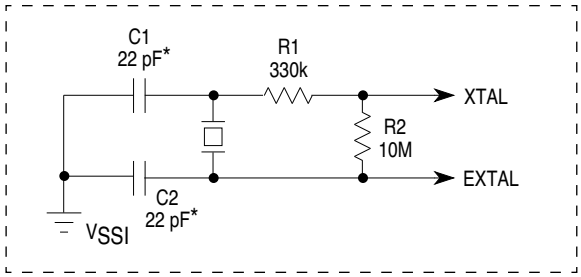


Figure 4-4 System Clock Block Diagram

4.3.1 Clock Sources

The state of the clock mode (MODCLK) pin during reset determines clock source. When MODCLK is held high during reset, the clock synthesizer generates a clock signal from either an internal or an external reference frequency — 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 — SYNCR control bits have no effect.

To generate a reference frequency using the internal oscillator a reference crystal must be connected between the EXTAL and XTAL pins. **Figure 4-5** 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.

32 OSCILLATOR

Figure 4-5 System Clock Oscillator Circuit

If an external reference signal or an external system clock signal is applied via the EXTAL pin, the XTAL pin must be left floating. External reference signal frequency must be less than or equal to maximum specified reference frequency. External system clock signal frequency must be less than or equal to maximum specified system clock frequency.

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:

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

4.3.2 Clock Synthesizer Operation

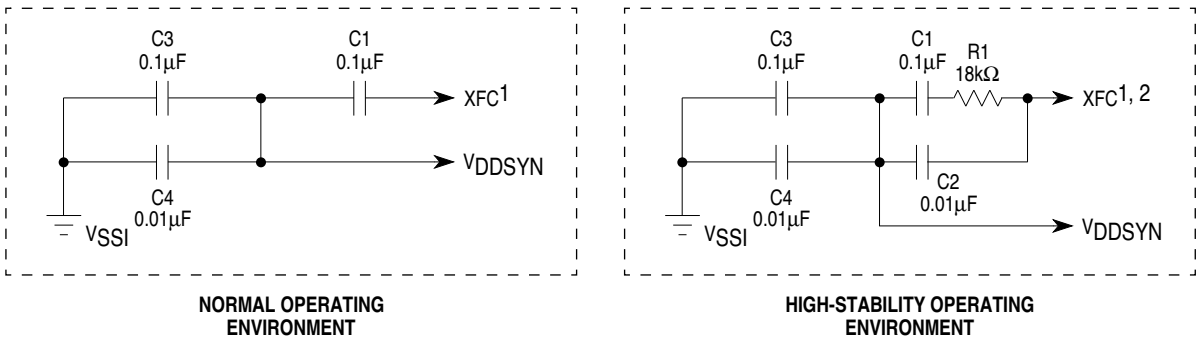
V_{DDSYN} is used to power the clock circuits when either an internal or an external reference frequency is applied. A separate power source increases MCU noise immunity and can be used to run the clock when the MCU is powered down. A quiet power sup-

ply 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, VCO frequency is either two or four times 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 a reference signal, either from the crystal oscillator or from an external source. 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 VCO output frequency.

Filter geometry can vary, depending upon the external environment and required clock stability. **Figure 4-6** shows two recommended filters. XFC pin leakage must be as specified in **APPENDIX A ELECTRICAL CHARACTERISTICS** 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.



1. Maintain low-leakage on the XFC node. See Appendix A electrical characteristics for more information.
2. Recommended loop filter for reduced sensitivity to low-frequency noise.

16/32 XFC CONN

Figure 4-6 System Clock Filter Networks

The synthesizer locks when VCO frequency is equal to EXTAL frequency. Lock time is affected by the filter time constant and by the amount of difference between the two comparator inputs. Whenever comparator input changes, the synthesizer must relock. Lock status is shown by the SLOCK bit in SYNCR. During power-up, the MCU does not come out of reset state until the synthesizer locks. Crystal type, characteristic frequency, and layout of external oscillator circuitry affect lock time.

When the clock synthesizer is used, control register SYNCR determines operating frequency and various modes of operation. The SYNCR W bit controls a three-bit prescaler in the feedback divider. Setting W increases VCO speed by a factor of four. The SYNCR Y field determines the count modulus for a modulo 64 down counter, causing it to divide by a value of Y + 1. When W or Y values change, VCO frequency changes, and there is a VCO relock delay. The SYNCR 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 system clock frequency is one-fourth VCO frequency; setting X disables the divider, doubling clock speed without changing VCO speed. There is no relock delay when clock speed is changed by the X bit.

Clock frequency is determined by SYNCR bit settings as follows:

$$F_{SYSTEM} = F_{REFERENCE}[4(Y + 1)(2^{2W + X})]$$

The reset state of SYNCR ($3F00) produces a modulus-64 count.

For the device to perform 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 in **APPENDIX A ELECTRICAL CHARACTERISTICS**.

Table 4-7 shows clock control multipliers for all possible combinations of SYNCR bits. **Table 4-8** shows clock frequencies available with a 32.768-kHz reference and a maximum specified clock frequency of 20.97 MHz.

Table 4-7 Clock Control Multipliers

To obtain clock frequency in kilohertz, find counter modulus in the left column, then multiply reference frequency by value in appropriate prescaler cell.

Modulus	Prescalers			
Y	[W:X] = 00	[W:X] = 01	[W:X] = 10	[W:X] = 11
000000	4	8	16	32
000001	8	16	32	64
000010	12	24	48	96
011111	16	32	64	128
000011	20	40	80	160
000100	24	48	96	192
000101	28	56	112	224
000110	32	64	128	256
000111	36	72	144	288
001000	40	80	160	320
001001	44	88	176	352
001010	48	96	192	384
001011	52	104	208	416
001100	56	112	224	448
001101	60	120	240	480
001110	64	128	256	512
001111	68	136	272	544
010000	72	144	288	576
010001	76	152	304	608

Table 4-7 Clock Control Multipliers (Continued)

To obtain clock frequency in kilohertz, find counter modulus in the left column, then multiply reference frequency by value in appropriate prescaler cell.

Modulus	Prescalers			
Y	[W:X] = 00	[W:X] = 01	[W:X] = 10	[W:X] = 11
010010	80	160	320	640
010011	84	168	336	672
010100	88	176	352	704
010101	92	184	368	736
010110	96	192	384	768
010111	100	200	400	800
011000	104	208	416	832
011001	108	216	432	864
011010	112	224	448	896
011011	116	232	464	928
011100	120	240	480	960
011101	124	248	496	992
011110	128	256	512	1024
100000	132	264	528	1056
100001	136	272	544	1088
100010	140	280	560	1120
100011	144	288	576	1152
100100	148	296	592	1184
100101	152	304	608	1216
100110	156	312	624	1248
100111	160	320	640	1280
101000	164	328	656	1312
101001	168	336	672	1344
101010	172	344	688	1376
101011	176	352	704	1408
101100	180	360	720	1440
101101	184	368	736	1472
101110	188	376	752	1504
101111	192	384	768	1536
110000	196	392	784	1568
110001	200	400	800	1600
110010	204	408	816	1632
110011	208	416	832	1664
110100	212	424	848	1696
110101	216	432	864	1728
110110	220	440	880	1760
110111	224	448	896	1792
111000	228	456	912	1824
111001	232	464	928	1856
111010	236	472	944	1888
111011	240	480	960	1920
111100	244	488	976	1952
111101	248	496	992	1984
111110	252	504	1008	2016
111111	256	512	1024	2048

Table 4-8 System Frequencies from 32.768–kHz Reference

To obtain clock frequency in kilohertz, find counter modulus in the left column, then multiply reference frequency by value in appropriate prescaler cell.

Modulus	Prescaler			
Y	[W:X] = 00	[W:X] = 01	[W:X] = 10	[W:X] = 11
000000	131	262	524	1049
000001	262	524	1049	2097
000010	393	786	1573	3146
000011	524	1049	2097	4194
000100	655	1311	2621	5243
000101	786	1573	3146	6291
000110	918	1835	3670	7340
000111	1049	2097	4194	8389
001000	1180	2359	4719	9437
001001	1311	2621	5243	10486
001010	1442	2884	5767	11534
001011	1573	3146	6291	12583
001100	1704	3408	6816	13631
001101	1835	3670	7340	14680
001110	1966	3932	7864	15729
001111	2097	4194	8389	16777
010000	2228	4456	8913	17826
010001	2359	4719	9437	18874
010010	2490	4981	9961	19923
010011	2621	5243	10486	20972
010100	2753	5505	11010	22020
010101	2884	5767	11534	23069
010110	3015	6029	12059	24117
010111	3146	6291	12583	25166
011000	3277	6554	13107	26214
011001	3408	6816	13631	27263
011010	3539	7078	14156	28312
011011	3670	7340	14680	29360
011100	3801	7602	15204	30409
011101	3932	7864	15729	31457
011110	4063	8126	16253	32506
011111	4194	8389	16777	33554
100000	4325	8651	17302	34603
100001	4456	8913	17826	35652
100010	4588	9175	18350	36700
100011	4719	9437	18874	37749
100100	4850	9699	19399	38797
100101	4981	9961	19923	39846
100110	5112	10224	20447	40894
100111	5243	10486	20972	41943
101000	5374	10748	21496	42992
101001	5505	11010	22020	44040
101010	5636	11272	22544	45089
101011	5767	11534	23069	46137
101100	5898	11796	23593	47186

Table 4-8 System Frequencies from 32.768–kHz Reference (Continued)

To obtain clock frequency in kilohertz, find counter modulus in the left column, then multiply reference frequency by value in appropriate prescaler cell.

Modulus	Prescaler			
Y	[W:X] = 00	[W:X] = 01	[W:X] = 10	[W:X] = 11
101101	6029	12059	24117	48234
101110	6160	12321	24642	49283
101111	6291	12583	25166	50332
110000	6423	12845	25690	51380
110001	6554	13107	26214	52428
110010	6685	13369	26739	53477
110011	6816	13631	27263	54526
110100	6947	13894	27787	55575
110101	7078	14156	28312	56623
110110	7209	14418	28836	57672
110111	7340	14680	29360	58720
111000	7471	14942	2988	59769
111001	7602	15204	30409	60817
111010	7733	15466	30933	61866
111011	7864	15729	31457	62915
111100	7995	15991	31982	63963
111101	8126	16253	32506	65011
111110	8258	16515	33030	66060
111111	8389	16777	33554	67109

4.3.3 External Bus Clock

The state of the external clock division bit (EDIV) in SYNCR determines clock rate for the external bus clock signal (ECLK) 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 **4.8 Chip Selects** for more information about the external bus clock.

4.3.4 Low-Power Operation

Low-power operation is initiated by the CPU32. To reduce power consumption selectively, the CPU can set the STOP bits in each module configuration register. To minimize overall microcontroller power consumption, the CPU can execute the LPSTOP instruction, which causes the SIM to turn off the system clock.

When individual module STOP bits are set, clock signals inside each module are turned off, but module registers are still accessible.

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 SIM brings the MCU out of low-power operation when either an interrupt of higher priority than the stored mask or a reset occurs. Refer to **4.5.4.2 LPSTOP Broadcast Cycle** and **SECTION 5 CENTRAL PROCESSING UNIT** for more information.

During a low-power stop, unless the system clock signal is supplied by an external source and that source is removed, the SIM clock control logic and the SIM 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. The SIM can also continue to generate the CLKOUT signal while in low-power mode.

The stop mode system integration module clock (STSIM) and stop mode external clock (STEXT) bits in SYNCR determine clock operation during low-power stop. **Table 4-9** is a summary of the effects of STSIM and STEXT. MODCLK value is the logic level on the MODCLK pin during the last reset before LPSTOP execution. Any clock in the off state is held low. If the synthesizer VCO is turned off during LPSTOP, there is a PLL relock delay after the VCO is turned back on.

Table 4-9 Clock Control

Mode	Pins		SYNCR Bits		Clock Status		
LPSTOP	MODCLK	EXTAL	STSIM	STEXT	SIMCLK	CLKOUT	ECLK
No	0	External Clock	X	X	External Clock	External Clock	External Clock
Yes	0	External Clock	0	0	External Clock	Off	Off
Yes	0	External Clock	0	1	External Clock	External Clock	External Clock
Yes	0	External Clock	1	0	External Clock	Off	Off
Yes	0	External Clock	1	1	External Clock	External Clock	External Clock
No	1	Crystal or Reference	X	X	VCO	VCO	VCO
Yes	1	Crystal or Reference	0	0	Crystal or Reference	Off	Off
Yes	1	Crystal or Reference	0	1	Crystal or Reference	Crystal/ Reference	Off
Yes	1	Crystal or Reference	1	0	VCO	Off	Off
Yes	1	Crystal or Reference	1	1	VCO	VCO	VCO

4.3.5 Loss of Reference Signal

The state of the reset enable (RSTEN) bit in SYNCR determines what happens when clock logic detects a reference failure.

When RSTEN is cleared (default state out of reset), the clock synthesizer is forced into an operating condition referred to as limp mode. Limp mode frequency varies from device to device, but maximum limp frequency does not exceed one half maximum system clock when X = 0, or maximum system clock frequency when X = 1.

When RSTEN is set, the SIM resets the MCU.

The limp status bit (SLIMP) in SYNCR indicates whether the synthesizer has a reference signal. It is set when a reference failure is detected.

4.4 External Bus Interface

The external bus interface (EBI) transfers information between the internal MCU bus and external devices. **Figure 4-7** shows a basic system with external memory and peripherals.

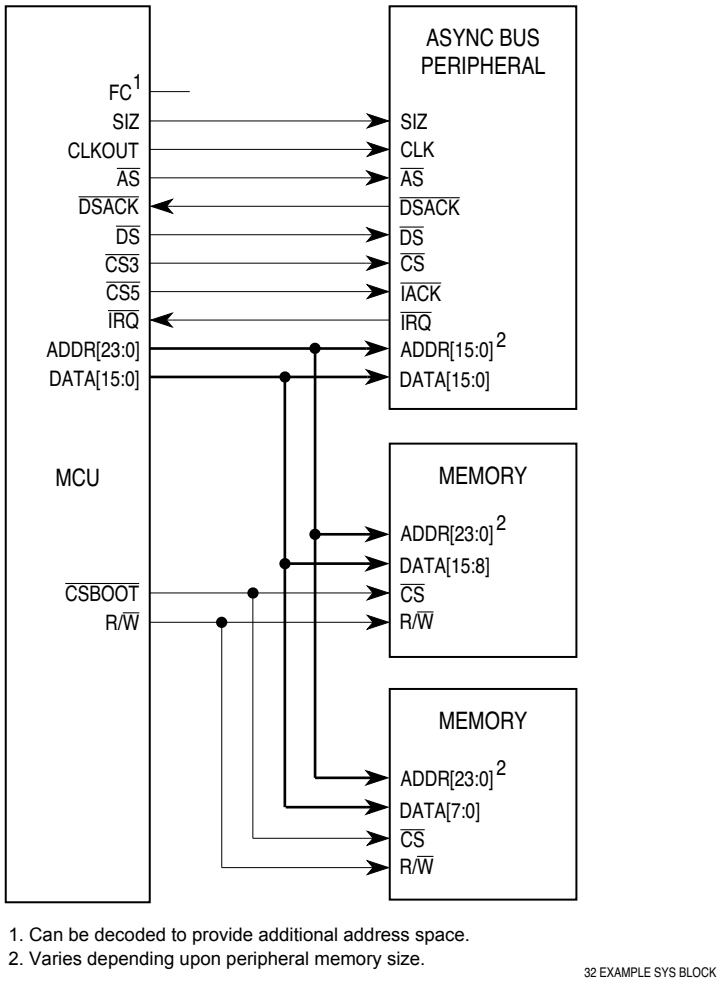


Figure 4-7 MCU Basic System

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 data transfer (SIZ1 and SIZ0) and data size acknowledge pins ($\overline{\text{DSACK1}}$ and $\overline{\text{DSACK0}}$). Multiple bus cycles may be required for a transfer to or from an 8-bit port.

The maximum number of bits transferred during an access is referred to as port width. Widths of eight and sixteen bits can be accessed by asynchronous bus cycles controlled by the data size (SIZ[1:0]) and the data and size acknowledge ($\overline{\text{DSACK[1:0]}}$) signals. Multiple bus cycles may be required for a dynamically-sized transfer.

To add flexibility and minimize the necessity for external logic, MCU chip-select logic can be synchronized with EBI transfers. Refer to **4.8 Chip Selects** for more information.

4.4.1 Bus Signals

The address bus provides addressing information to external devices. The data bus transfers 8-bit and 16-bit data between the MCU and external devices. 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.

Control signals indicate the beginning of each bus cycle, the address space it is to take place in, the size of the transfer, and the type of cycle. External devices decode these signals and respond to transfer data and terminate the bus cycle. The EBI operates in an asynchronous mode for any port width.

4.4.1.1 Address Bus

Bus signals ADDR[23:0] define the address of the byte (or 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.

4.4.1.2 Address Strobe

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

4.4.1.3 Data Bus

Signals DATA[15:0] form a bidirectional, nonmultiplexed parallel bus that transfers data to or from the MCU. A read or write operation can transfer eight or sixteen 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.

4.4.1.4 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.

4.4.1.5 Read/Write Signal

The read/write signal (R/$\overline{W}$) 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 $\overline{AS}$ is asserted. R/$\overline{W}$ only transitions when a write cycle is preceded by a read cycle or vice versa. The signal may remain low for two consecutive write cycles.

4.4.1.6 Size Signals

Size signals (SIZ[1:0]) indicate the number of bytes remaining to be transferred during an operand cycle. They are valid while the address strobe ($\overline{AS}$) is asserted. **Table 4-10** shows SIZ0 and SIZ1 encoding.

Table 4-10 Size Signal Encoding

SIZ1	SIZ0	Transfer Size
0	1	Byte
1	0	Word
1	1	3 Byte
0	0	Long Word

4.4.1.7 Function Codes

The CPU generates function code output signals FC[2:0] to indicate the type of activity occurring on the data or address bus. These signals can be considered address extensions that can be externally decoded to determine which of eight external address spaces is accessed during a bus cycle.

Address space 7 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 4-11 shows address space encoding.

Table 4-11 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

The supervisor bit in the status register determines whether the CPU is operating in supervisor or user mode. Addressing mode and the instruction being executed determine whether a memory access is to program or data space.

4.4.1.8 Data and Size Acknowledge Signals

During normal bus transfers, external devices assert the data and size acknowledge signals ($\overline{DSACK[1:0]}$) to indicate port width to the MCU. During a read cycle, these 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 terminate. $\overline{DSACK[1:0]}$ can also be supplied internally by chip-select logic. Refer to **4.8 Chip Selects** for more information.

4.4.1.9 Bus Error Signal

The bus error signal ($\overline{BERR}$) is asserted when a bus cycle is not properly terminated by $\overline{DSACK}$ or $\overline{AVEC}$ assertion. $\overline{BERR}$ can also be asserted at the same time as $\overline{DSACK}$, provided the appropriate timing requirements are met. Refer to **4.5.5 Bus Exception Control Cycles** for more information.

The internal bus monitor can generate the $\overline{BERR}$ signal for internal and internal-to-external transfers. An external bus master must provide its own $\overline{BERR}$ generation and drive the $\overline{BERR}$ pin, because the internal $\overline{BERR}$ monitor has no information about transfers initiated by an external bus master. Refer to **4.5.6 External Bus Arbitration** for more information.

4.4.1.10 Halt Signal

The halt signa ($\overline{HALT}$) can be asserted by an external device for debugging purposes to cause single bus cycle operation or (in combination with $\overline{BERR}$) a retry of a bus cycle in error. The $\overline{HALT}$ signal affects external bus cycles only, so a program not requiring the use of external bus may continue executing, unaffected by the $\overline{HALT}$ signal. When the MCU completes a bus cycle with the $\overline{HALT}$ signal asserted, DATA[15:0] is placed in the high-impedance state, and bus control signals are driven inactive; the address, function code, size, and read/write signals remain in the same state. If $\overline{HALT}$ is still asserted once bus mastership is returned to the MCU, the address, function code, size, and read/write signals are again driven to their previous states. The MCU does not service interrupt requests while it is halted. Refer to **4.5.5 Bus Exception Control Cycles** for further information.

4.4.1.11 Autovector Signal

The autovector signal ($\overline{AVEC}$) can be used to terminate external interrupt acknowledge cycles. Assertion of $\overline{AVEC}$ causes the CPU32 to generate vector numbers to locate an interrupt handler routine. If it is continuously asserted, autovectors are generated for all external interrupt requests. $\overline{AVEC}$ is ignored during all other bus cycles. Refer to **4.7 Interrupts** for more information. $\overline{AVEC}$ for external interrupt requests can also be supplied internally by chip-select logic. Refer to **4.8 Chip Selects** for more information. The autovector function is disabled when there is an external bus master. Refer to **4.5.6 External Bus Arbitration** for more information.

4.4.2 Dynamic Bus Sizing

The MCU dynamically interprets the port size of an addressed device during each bus cycle, allowing operand transfers to or from 8-bit and 16-bit ports.

During an operand transfer cycle, an external device signals its port size and indicates completion of the bus cycle to the MCU through the use of the $\overline{DSACK}$ inputs, as shown in **Table 4-12**. Chip-select logic can generate data and size acknowledge signals for an external device. Refer to **4.8 Chip Selects** for further information.

Table 4-12 Effect of $\overline{\text{DSACK}}$ Signals

$\overline{\text{DSACK1}}$	$\overline{\text{DSACK0}}$	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

If the CPU 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 $\overline{\text{DSACK}}$ signals to indicate the port width. For instance, a 16-bit device always returns $\overline{\text{DSACK}}$ 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 4-8**. OP[0:3] represent the order of access. For instance, OP0 is the most significant byte of a long-word operand, and is accessed first, while OP3, the least significant byte, is accessed last. The two bytes of a word-length operand are OP0 (most significant) and OP1. The single byte of a byte-length operand is OP0.

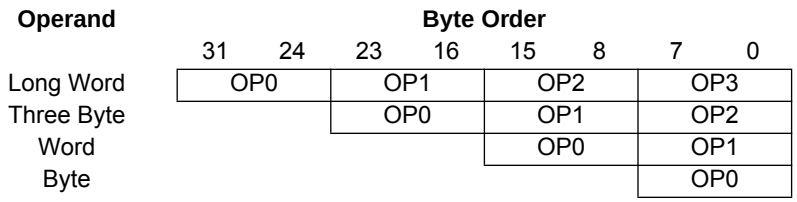


Figure 4-8 Operand Byte Order

4.4.3 Operand Alignment

The EBI 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.

4.4.4 Misaligned Operands

CPU32 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 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 through a 16-bit port, the most significant operand word is transferred on the first bus cycle and the least significant operand word is transferred on a following bus cycle.

4.4.5 Operand Transfer Cases

Table 4-13 is a summary of 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. The following paragraphs discuss all the allowable transfer cases in detail.

Table 4-13 Operand Transfer Cases

					Read Cycles		Write Cycles		
Num	Transfer Case	SIZ [1:0]	ADDR0	DSACK [1:0]	DATA [15:8]	DATA [7:0]	DATA [15:8]	DATA [7:0]	Next Cycle
1	Byte to 8-Bit Port (Even/Odd)	01	X	10	OP0	—	OP0	(OP0)	—
2	Byte to 16-Bit Port (Even)	01	0	01	OP0	—	OP0	(OP0)	—
3	Byte to 16-Bit Port (Odd)	01	1	01	—	OP0	(OP0)	OP0	—
4	Word to 8-Bit Port (Aligned)	10	0	10	OP0	—	OP0	(OP1)	1
5	Word to 8-Bit Port (Misaligned)[1]	10	1	10	OP0	—	OP0	(OP0)	1
6	Word to 16-Bit Port (Aligned)	10	0	11	OP0	OP1	OP0	OP1	—
7	Word to 16-Bit Port (Misaligned)[1]	10	1	01	—	OP0	(OP0)	OP0	2
8	Long Word to 8-Bit Port (Aligned)	00	0	10	OP0	—	OP0	(OP1)	13
9	Long Word to 8-Bit Port (Misaligned)[1]	10	1	10	OP0	—	OP0	(OP0)	12
10	Long Word to 16-Bit Port (Aligned)	00	0	01	OP0	OP1	OP0	OP1	6
11	Long Word to 16-Bit Port (Misaligned)[1]	10	1	01	—	OP0	(OP0)	OP0	2
12	3 Byte to 8-Bit Port (Aligned)[2]	11	0	10	OP0	—	OP0	(OP1)	5
13	3 Byte to 8-Bit Port (Misaligned)[2]	11	1	10	OP0	—	OP0	(OP0)	4

NOTES:
1. The CPU32 does not support misaligned transfers.
2. Three-byte transfer cases occur only as a result of a long word to byte transfer.

4.5 Bus Operation

Internal microcontroller modules are typically accessed in two system clock cycles, with no wait states. Regular external bus cycles use handshaking between the MCU and external peripherals to manage transfer size and data. These accesses take three system clock cycles, again with no wait states. During regular cycles, wait states can be inserted as needed by bus control logic. Refer to **4.5.2 Regular Bus Cycles** for more information.

Fast-termination cycles, which are two-cycle external accesses with no wait states, use chip-select logic to generate handshaking signals internally. Chip-select logic can also be used to insert wait states before internal generation of handshaking signals. Refer to **4.5.3 Fast Termination Cycles** and **4.8 Chip Selects** for more information. Bus control signal timing, as well as chip-select signal timing, are specified in **APPENDIX A ELECTRICAL CHARACTERISTICS**. Refer to the *SIM Reference Manual* (SIMRM/AD) for more information about each type of bus cycle.

The MCU is responsible for de-skewing signals it issues at both the start and the end of a cycle. In addition, the MCU is responsible for de-skewing acknowledge and data signals from peripheral devices.

4.5.1 Synchronization to CLKOUT

External devices connected to the MCU bus can operate at a clock frequency different from the frequencies of the MCU as long as the external devices satisfy the interface signal timing constraints. Although bus cycles are classified as asynchronous, they are interpreted relative to the MCU system clock output (CLKOUT).

Descriptions are made in terms of individual system clock states, labeled {S0, S1, S2,..., SN}. The designation “state” refers to the logic level of the clock signal, and does not correspond to any implemented machine state. A clock cycle consists of two successive states. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for more information.

Bus cycles terminated by $\overline{\text{DSACK}}$ assertion normally require a minimum of three CLKOUT cycles. To support systems that use CLKOUT to generate $\overline{\text{DSACK}}$ and other inputs, asynchronous input setup time and asynchronous input hold times are specified. When these specifications are met, the MCU is guaranteed to recognize the appropriate signal on a specific edge of the CLKOUT signal.

For a read cycle, when assertion of $\overline{\text{DSACK}}$ is recognized on a particular falling edge of the clock, valid data is latched into the MCU on the next falling clock edge, provided that the data meets the data setup time. In this case, the parameter for asynchronous operation can be ignored.

When a system asserts $\overline{\text{DSACK}}$ for the required window around the falling edge of S2 and obeys the bus protocol by maintaining $\overline{\text{DSACK}}$ and $\overline{\text{BERR}}$ or $\overline{\text{HALT}}$ until and throughout the clock edge that negates $\overline{\text{AS}}$, no wait states are inserted. The bus cycle runs at the maximum speed of three clocks per cycle.

To ensure proper operation in a system synchronized to CLKOUT, when either $\overline{\text{BERR}}$, or $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ is asserted after $\overline{\text{DSACK}}$, $\overline{\text{BERR}}$ (or $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$) assertion must satisfy the appropriate data-in setup and hold times before the falling edge of the clock cycle after $\overline{\text{DSACK}}$ is recognized.

4.5.2 Regular Bus Cycles

The following paragraphs contain a discussion of cycles that use external bus control logic. Refer to **4.5.3 Fast Termination Cycles** for information about fast cycles.

To initiate a transfer, the MCU asserts an address and the SIZ[1:0] signals. The SIZ signals and ADDR0 are externally decoded to select the active portion of the data bus (refer to **4.4.2 Dynamic Bus Sizing**). When $\overline{AS}$, $\overline{DS}$, and R/$\overline{W}$ are valid, a peripheral device either places data on the bus (read cycle) or latches data from the bus (write cycle), then asserts a $\overline{DSACK}$[1:0] combination that indicates port size.

The $\overline{DSACK}$[1:0] signals can be asserted before the data from a peripheral device is valid on a read cycle. To ensure valid data is latched into the MCU, a maximum period between $\overline{DSACK}$ assertion and $\overline{DS}$ assertion is specified.

There is no specified maximum for the period between the assertion of $\overline{AS}$ and $\overline{DSACK}$. Although the MCU can transfer data in a minimum of three clock cycles when the cycle is terminated with $\overline{DSACK}$, the MCU inserts wait cycles in clock period increments until either $\overline{DSACK}$ signal goes low.

NOTE

The SIM bus monitor asserts $\overline{BERR}$ when response time exceeds a predetermined limit. Bus monitor period is determined by the BMT field in SYPCR. The bus monitor cannot be disabled; maximum monitor period is 64 system clock cycles.

If no peripheral responds to an access, or if an access is invalid, external logic should assert the $\overline{BERR}$ or $\overline{HALT}$ signals to abort the bus cycle (when $\overline{BERR}$ and $\overline{HALT}$ are asserted simultaneously, the CPU32 acts as though only $\overline{BERR}$ is asserted). If bus termination signals are not asserted within a specified period, the bus monitor terminates the cycle.

4.5.2.1 Read Cycle

During a read cycle, the MCU transfers data from an external memory or peripheral device. If the instruction specifies a long-word or word operation, the MCU attempts to read two bytes at once. For a byte operation, the MCU reads one byte. The portion of the data bus from which each byte is read depends on operand size, peripheral address, and peripheral port size. **Figure 4-9** is a flowchart of a word read cycle. Refer to **4.4.2 Dynamic Bus Sizing**, **4.4.4 Misaligned Operands**, and the *SIM Reference Manual* (SIMRM/AD) for more information.

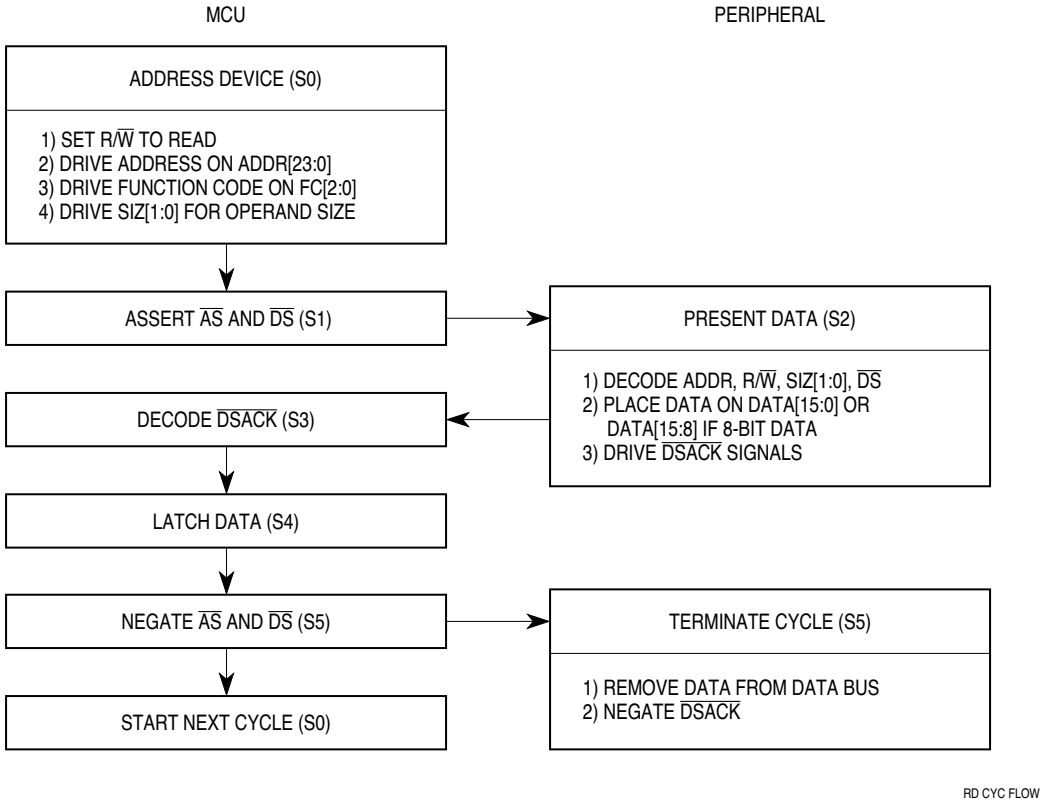


Figure 4-9 Word Read Cycle Flowchart

4.5.2.2 Write Cycle

During a write cycle, the MCU transfers data to an external memory or peripheral device. If the instruction specifies a long-word or word operation, the MCU attempts to write two bytes at once. For a byte operation, the MCU writes one byte. The portion of the data bus upon which each byte is written depends on operand size, peripheral address, and peripheral port size.

Refer to **4.4.2 Dynamic Bus Sizing** and **4.4.4 Misaligned Operands** for more information. **Figure 4-10** is a flowchart of a write-cycle operation for a word transfer. Refer to the *SIM Reference Manual* (SIMRM/AD) for more information.

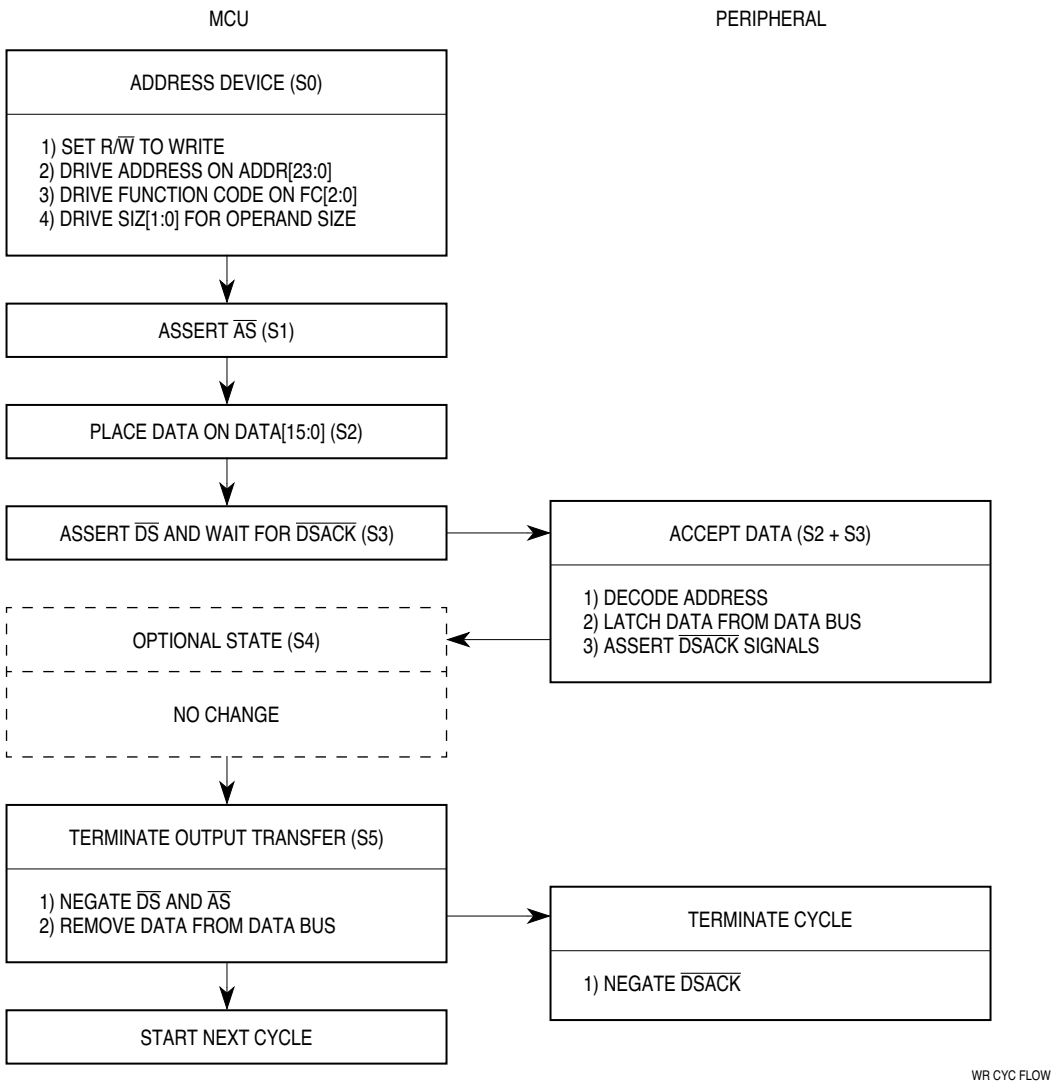


Figure 4-10 Write Cycle Flowchart

4.5.3 Fast Termination Cycles

When an external device has a fast access time, the chip-select circuit fast-termination option can provide a two-cycle external bus transfer. Because the chip-select circuits are driven from the system clock, the bus cycle termination is inherently synchronized with the system clock.

If multiple chip selects are to be used to select the same device that can support fast termination, and match conditions can occur simultaneously, program the DSACK field in each associated chip-select option register for fast termination. Alternately, program one DSACK field for fast termination and the remaining DSACK fields for external termination.

Fast termination cycles use internal handshaking signals generated by the chip-select logic. To initiate a transfer, the MCU asserts an address and the SIZ[1:0] signals.

When AS, DS, and R/W are valid, a peripheral device either places data on the bus (read cycle) or latches data from the bus (write cycle). At the appropriate time, chip-select logic asserts data and size acknowledge signals.

The DSACK option fields in the chip-select option registers determine whether internally generated DSACK or externally generated DSACK are used. For fast termination cycles, the F-term encoding (%1110) must be used. Refer to **4.8.1 Chip-Select Registers** for information about fast-termination setup.

To use fast-termination, an external device must be fast enough to have data ready, within the specified setup time, by the falling edge of S4. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for tabular information about fast termination timing.

When fast termination is in use, DS is asserted during read cycles but not during write cycles. The STRB field in the chip-select option register used must be programmed with the address strobe encoding to assert the chip select signal for a fast-termination write.

4.5.4 CPU Space Cycles

Function code signals FC[2:0] designate which of eight external address spaces is accessed during a bus cycle. Address space 7 is designated CPU space. CPU space is used for control information not normally associated with read or write bus cycles. Function codes are valid only while AS is asserted. Refer to **4.4.1.7 Function Codes** for more information on codes and encoding.

During a CPU space access, ADDR[19:16] are encoded to reflect the type of access being made. **Figure 4-11** shows the three encodings used by 68300 family microcontrollers. These encodings represent breakpoint acknowledge (Type $0) cycles, low power stop broadcast (Type $3) cycles, and interrupt acknowledge (Type $F) cycles. Refer to **4.7 Interrupts** for information about interrupt acknowledge bus cycles.

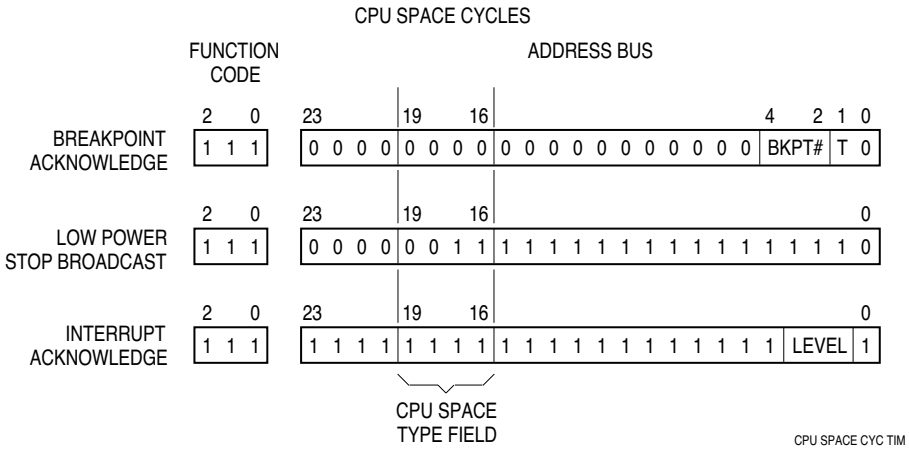


Figure 4-11 CPU Space Address Encoding

4.5.4.1 Breakpoint Acknowledge Cycle

Breakpoints stop program execution at a predefined point during system development. Breakpoints can be used alone or in conjunction with the background debugging mode. The following paragraphs discuss breakpoint processing when background debugging mode is not enabled. See **SECTION 5 CENTRAL PROCESSING UNIT** for more information on exception processing and the background debugging mode.

In M68300 microcontrollers, both hardware and software can initiate breakpoints.

4.5.4.1.1 Software Breakpoints

The CPU32 $\overline{\text{BKPT}}$ instruction allows the user to insert breakpoints through software. The CPU responds to this instruction by initiating a breakpoint-acknowledge read cycle in CPU space. It places the breakpoint acknowledge (%0000) code on ADDR[19:16], the breakpoint number (bits [2:0] of the BKPT opcode) in ADDR[4:2], and %0 (indicating a software breakpoint) on ADDR1.

The external breakpoint circuitry decodes the function code and address lines and responds by either asserting $\overline{\text{BERR}}$ or placing an instruction word on the data bus and asserting $\overline{\text{DSACK}}$.

If the bus cycle is terminated by $\overline{\text{DSACK}}$, the CPU32 reads the instruction on the data bus and inserts the instruction into the pipeline. (For 8-bit ports, this instruction fetch may require two read cycles.)

If the bus cycle is terminated by $\overline{\text{BERR}}$, the CPU32 then performs illegal-instruction exception processing: it acquires the number of the illegal-instruction exception vector, computes the vector address from this number, loads the content of the vector address into the PC, and jumps to the exception handler routine at that address.

4.5.4.1.2 Hardware Breakpoints

Assertion of the $\overline{\text{BKPT}}$ input initiates a hardware breakpoint. The CPU responds by initiating a breakpoint-acknowledge read cycle in CPU space. It places $00001E on the address bus. (The breakpoint acknowledge code of %0000 is placed on ADDR[19:16], the breakpoint number value of %111 is placed on ADDR[4:2], and ADDR1 is set to one, indicating a hardware breakpoint.)

The external breakpoint circuitry decodes the function code and address lines, places an instruction word on the data bus, and asserts $\overline{\text{BERR}}$. The CPU then performs hardware breakpoint exception processing: it acquires the number of the hardware breakpoint exception vector, computes the vector address from this number, loads the content of the vector address into the PC, and jumps to the exception handler routine at that address. If the external device asserts $\overline{\text{DSACK}}$ rather than $\overline{\text{BERR}}$, the CPU ignores the breakpoint and continues processing.

When $\overline{\text{BKPT}}$ assertion is synchronized with an instruction prefetch, processing of the breakpoint exception occurs at the end of that instruction. The prefetched instruction is “tagged” with the breakpoint when it enters the instruction pipeline, and the breakpoint exception occurs after the instruction executes. If the pipeline is flushed before

the tagged instruction is executed, no breakpoint occurs. When $\overline{\text{BKPT}}$ assertion is synchronized with an operand fetch, exception processing occurs at the end of the instruction during which $\overline{\text{BKPT}}$ is latched.

Refer to the *CPU32 Reference Manual* (CPU32RM/AD) and the *SIM Reference Manual* (SIMRM/AD) for additional information.

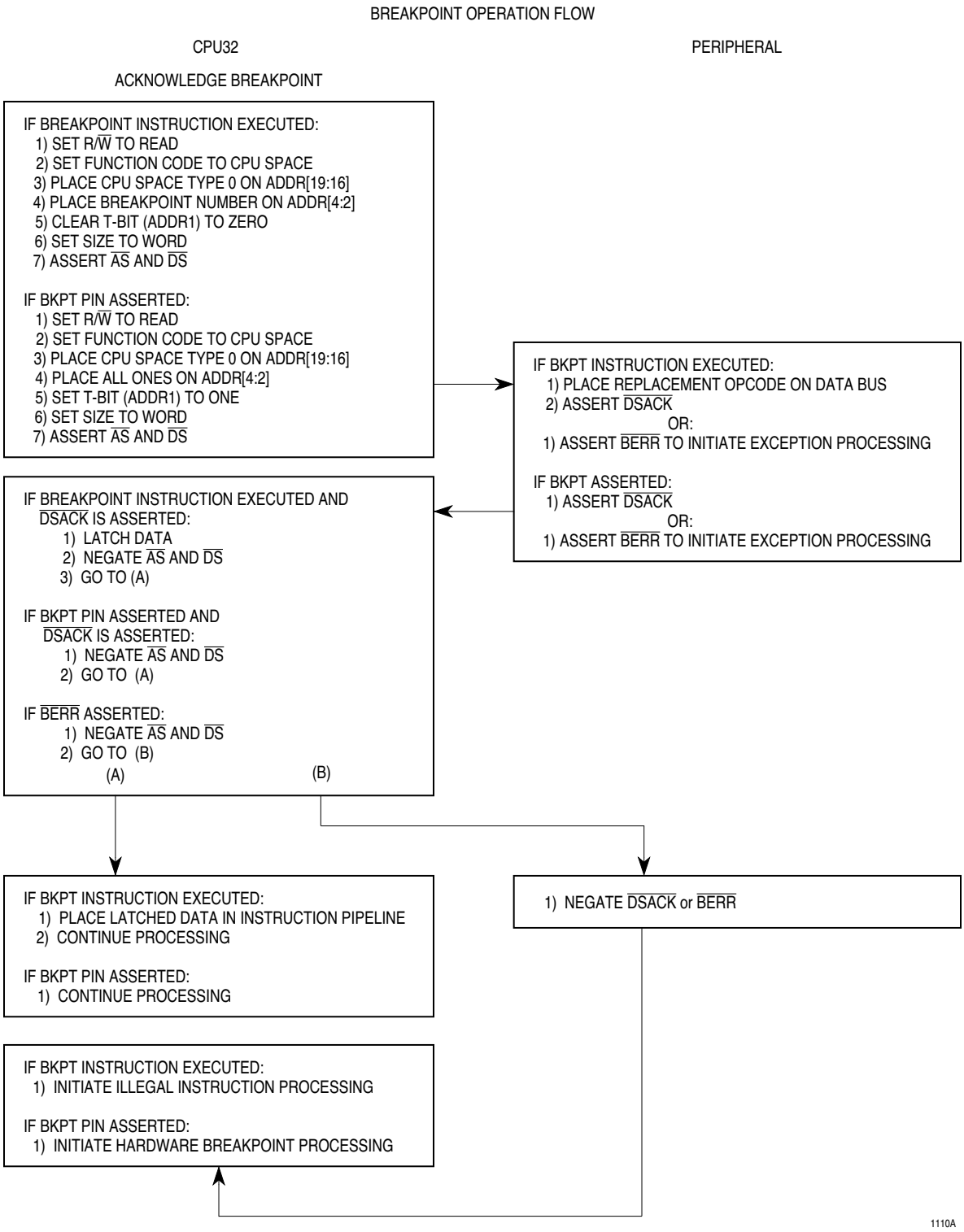


Figure 4-12 Breakpoint Operation Flowchart

4.5.4.2 LPSTOP Broadcast Cycle

Low-power stop is initiated by the CPU32. Individual modules can be stopped by setting the STOP bits in each module configuration register, or the SIM can turn off system clocks after execution of the LPSTOP instruction. When the CPU executes LPSTOP, the LPSTOP broadcast cycle is generated. The SIM brings the MCU out of low-power mode when either an interrupt of higher priority than the stored mask or a reset occurs. Refer to and **SECTION 5 CENTRAL PROCESSING UNIT** for more information.

During an LPSTOP broadcast cycle, the CPU performs a CPU space write to address $3FFFE. This write puts a copy of the interrupt mask value in the clock control logic. The mask is encoded on the data bus as shown in **Figure 4-13**. The LPSTOP CPU space cycle is shown externally (if the bus is available) as an indication to external devices that the MCU is going into low-power stop mode. The SIM provides an internally generated $\overline{\text{DSACK}}$ response to this cycle. The timing of this bus cycle is the same as for a fast write cycle.

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

Figure 4-13 LPSTOP Interrupt Mask Level

4.5.5 Bus Exception Control Cycles

An external device or a chip-select circuit must assert at least one of the $\overline{\text{DSACK}}$[1:0] signals or the $\overline{\text{AVEC}}$ signal to terminate a bus cycle normally. Bus error processing occurs when bus cycles are not terminated in the expected manner. The internal bus monitor can be used to generate $\overline{\text{BERR}}$ internally, causing a bus error exception to be taken. Bus cycles can also be terminated by assertion of the external $\overline{\text{BERR}}$ or $\overline{\text{HALT}}$ signal, or by assertion of the two signals simultaneously.

Acceptable bus cycle termination sequences are summarized as follows. The case numbers refer to **Table 4-5**, which indicates the results of each type of bus cycle termination.

Normal Termination

$\overline{\text{DSACK}}$ is asserted; $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ remain negated (case 1).

Halt Termination

$\overline{\text{HALT}}$ is asserted at the same time or before $\overline{\text{DSACK}}$, and $\overline{\text{BERR}}$ remains negated (case 2).

Bus Error Termination

$\overline{\text{BERR}}$ is asserted in lieu of, at the same time as, or before $\overline{\text{DSACK}}$, or after $\overline{\text{DSACK}}$, and $\overline{\text{HALT}}$ remains negated; $\overline{\text{BERR}}$ is negated at the same time or after $\overline{\text{DSACK}}$.

Retry Termination

HALT and BERR are asserted in lieu of, at the same time as, or before DSACK or after DSACK; BERR is negated at the same time or after DSACK; HALT may be negated at the same time or after BERR.

Table 4-14 shows various combinations of control signal sequences and the resulting bus cycle terminations.

Table 4-14 DSACK, BERR, and HALT Assertion Results

Case Number	Control Signal	Asserted on Rising Edge of State		Result
		N	N + 2	
1	DSACK BERR HALT	A NA NA	S NA X	Normal termination.
2	DSACK BERR HALT	A NA A/S	S NA S	Halt termination: normal cycle terminate and halt. Continue when HALT is negated.
3	DSACK BERR HALT	NA/A A NA	X S X	Bus error termination: terminate and take bus error exception, possibly deferred.
4	DSACK BERR HALT	A A NA	X S NA	Bus error termination: terminate and take bus error exception, possibly deferred.
5	DSACK BERR HALT	NA/A A A/S	X S S	Retry termination: terminate and retry when HALT is negated.
6	DSACK BERR HALT	A NA NA	X A A	Retry termination: terminate and retry when HALT is negated.

NOTES:
- N = The number of current even bus state (S2, S4, etc.).
- A = Signal is asserted in this bus state.
- NA = Signal is not asserted in this state
- X = Don't care.
- S = Signal was asserted in previous state and remains asserted in this state.

To properly control termination of a bus cycle for a retry or a bus error condition, DSACK, BERR, and HALT must be asserted and negated with the rising edge of the MCU clock. This ensures that when two signals are asserted simultaneously, the required setup time and hold time for both of them are met for the same falling edge of the MCU clock. (Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for timing requirements.) External circuitry that provides these signals must be designed with these constraints in mind, or else the internal bus monitor must be used.

DSACK, BERR, and HALT may be negated after AS is negated.

WARNING

If DSACK or BERR remain asserted into S2 of the next bus cycle, that cycle may be terminated prematurely.

4.5.5.1 Bus Errors

The CPU32 treats bus errors as a type of exception. Bus error exception processing begins when the CPU detects assertion of the IMB $\overline{\text{BERR}}$ signal (by the internal bus monitor or an external source) while the $\overline{\text{HALT}}$ signal remains negated.

$\overline{\text{BERR}}$ assertions do not force immediate exception processing. The signal is synchronized with normal bus cycles and is latched into the CPU32 at the end of the bus cycle in which it was asserted. Because bus cycles can overlap instruction boundaries, bus error exception processing may not occur at the end of the instruction in which the bus cycle begins. Timing of $\overline{\text{BERR}}$ detection/acknowledge is dependent upon several factors:

- Which bus cycle of an instruction is terminated by assertion of $\overline{\text{BERR}}$.
- The number of bus cycles in the instruction during which $\overline{\text{BERR}}$ is asserted.
- The number of bus cycles in the instruction following the instruction in which $\overline{\text{BERR}}$ is asserted.
- Whether $\overline{\text{BERR}}$ is asserted during a program space access or a data space access.

Because of these factors, it is impossible to predict precisely how long after occurrence of a bus error the bus error exception is processed.

CAUTION

The external bus interface does not latch data when an external bus cycle is terminated by a bus error. When this occurs during an instruction prefetch, the IMB precharge state (bus pulled high, or $FF) is latched into the CPU32 instruction register, with indeterminate results.

4.5.5.2 Double Bus Faults

Exception processing for bus error exceptions follows the standard exception processing sequence. Refer to **SECTION 5 CENTRAL PROCESSING UNIT** for more information about exceptions. However, a special case of bus error, called double bus fault, can abort exception processing.

$\overline{\text{BERR}}$ assertion is not detected until an instruction is complete. The $\overline{\text{BERR}}$ latch is cleared by the first instruction of the $\overline{\text{BERR}}$ exception handler. Double bus fault occurs in two ways:

1. When bus error exception processing begins and a second $\overline{\text{BERR}}$ is detected before the first instruction of the first exception handler is executed.
2. When one or more bus errors occur before the first instruction after a RESET exception is executed.
3. A bus error occurs while the CPU32 is loading information from a bus error stack frame during a return from exception (RTE) instruction.

Multiple bus errors within a single instruction that can generate multiple bus cycles cause a single bus error exception after the instruction has been executed.

Immediately after assertion of a second $\overline{\text{BERR}}$, the MCU halts and drives the $\overline{\text{HALT}}$ line low. Only a reset can restart a halted MCU. However, bus arbitration can still occur (refer to **4.5.6 External Bus Arbitration**). A bus error or address error that occurs after exception processing has been completed (during the execution of the exception handler routine, or later) does not cause a double bus fault. The MCU continues to retry the same bus cycle as long as the external hardware requests it.

4.5.5.3 Retry Operation

When an external device asserts $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ during a bus cycle, the MCU enters the retry sequence. A delayed retry can also occur. The MCU terminates the bus cycle, places the $\overline{\text{AS}}$ and $\overline{\text{DS}}$ signals in their inactive state, and does not begin another bus cycle until the $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ signals are negated by external logic. After a synchronization delay, the MCU retries the previous cycle using the same address, function codes, data (for a write), and control signals. The $\overline{\text{BERR}}$ signal should be negated before S2 of the read cycle to ensure correct operation of the retried cycle.

If $\overline{\text{BR}}$, $\overline{\text{BERR}}$, and $\overline{\text{HALT}}$ are all asserted on the same cycle, the EBI will enter the rerun sequence but first relinquishes the bus to an external master. Once the external master returns the bus and negates $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$, the EBI runs the previous bus cycle. This feature allows an external device to correct the problem that caused the bus error and then try the bus cycle again.

The MCU retries any read or write cycle of an indivisible read-modify-write operation separately; $\overline{\text{RMC}}$ remains asserted during the entire retry sequence. The MCU will not relinquish the bus while $\overline{\text{RMC}}$ is asserted. Any device that requires the MCU to give up the bus and retry a bus cycle during a read-modify-write cycle must assert $\overline{\text{BERR}}$ and $\overline{\text{BR}}$ only ($\overline{\text{HALT}}$ must remain negated). The bus error handler software should examine the read-modify-write bit in the special status word and take the appropriate action to resolve this type of fault when it occurs.

4.5.5.4 Halt Operation

When $\overline{\text{HALT}}$ is asserted while $\overline{\text{BERR}}$ is not asserted, the MCU halts external bus activity after negation of $\overline{\text{DSACK}}$. The MCU may complete the current word transfer in progress. For a long-word to byte transfer, this could be after S2 or S4. For a word to byte transfer, activity ceases after S2.

Negating and reasserting $\overline{\text{HALT}}$ according to timing requirements provides single-step (bus cycle to bus cycle) operation. The $\overline{\text{HALT}}$ signal affects external bus cycles only, so that a program that does not use external bus can continue executing. During dynamically-sized 8-bit transfers, external bus activity may not stop at the next cycle boundary. Occurrence of a bus error while $\overline{\text{HALT}}$ is asserted causes the CPU32 to initiate a retry sequence.

When the MCU completes a bus cycle while the $\overline{\text{HALT}}$ signal is asserted, the data bus goes to high-impedance state and the $\overline{\text{AS}}$ and $\overline{\text{DS}}$ signals are driven to their inactive states. Address, function code, size, and read/write signals remain in the same state.

The halt operation has no effect on bus arbitration (refer to **4.5.6 External Bus Arbitration**). However, when external bus arbitration occurs while the MCU is halted, address and control signals go to high-impedance state. If $\overline{\text{HALT}}$ is still asserted when the MCU regains control of the bus, address, function code, size, and read/write signals revert to the previous driven states. The MCU cannot service interrupt requests while halted.

4.5.6 External Bus Arbitration

MCU bus design provides for a single bus master at any one time. Either the MCU or an external device can be master. Bus arbitration protocols determine when an external device can become bus master. Bus arbitration requests are recognized during normal processing, $\overline{\text{HALT}}$ assertion, and when the CPU has halted due to a double bus fault.

The bus controller in the MCU manages bus arbitration signals so that the MCU has the lowest priority. External devices that need to obtain the bus must assert bus arbitration signals in the sequences described in the following paragraphs.

Systems that include several devices that can become bus master require external circuitry to assign priorities to the devices, so that when two or more external devices attempt to become bus master at the same time, the one having the highest priority becomes bus master first. The protocol sequence is:

A. An external device asserts bus request signal ($\overline{\text{BR}}$);
B. The MCU asserts the bus grant signal ($\overline{\text{BG}}$) to indicate that the bus is available;
C. An external device asserts the bus grant acknowledge ($\overline{\text{BGACK}}$) signal to indicate that it has assumed bus mastership.

$\overline{\text{BR}}$ can be asserted during a bus cycle or between cycles. $\overline{\text{BG}}$ is asserted in response to $\overline{\text{BR}}$. To guarantee operand coherency, $\overline{\text{BG}}$ is only asserted at the end of operand transfer. Additionally, $\overline{\text{BG}}$ is not asserted until the end of an indivisible read-modify-write operation (when $\overline{\text{RMC}}$ is negated).

If more than one external device can be bus master, required external arbitration must begin when a requesting device receives $\overline{\text{BG}}$. An external device must assert $\overline{\text{BGACK}}$ when it assumes mastership, and must maintain $\overline{\text{BGACK}}$ assertion as long as it is bus master.

Two conditions must be met for an external device to assume bus mastership. The device must receive $\overline{\text{BG}}$ through the arbitration process, and $\overline{\text{BGACK}}$ must be inactive, indicating that no other bus master is active. This technique allows the processing of bus requests during data transfer cycles.

$\overline{\text{BG}}$ is negated a few clock cycles after $\overline{\text{BGACK}}$ transition. However, if bus requests are still pending after $\overline{\text{BG}}$ is negated, the MCU asserts $\overline{\text{BG}}$ again within a few clock cycles. This additional $\overline{\text{BG}}$ assertion allows external arbitration circuitry to select the next bus master before the current master has released the bus.

Refer to **Figure 4-14**, which shows bus arbitration for a single device. The flowchart shows $\overline{\text{BR}}$ negated at the same time $\overline{\text{BGACK}}$ is asserted.

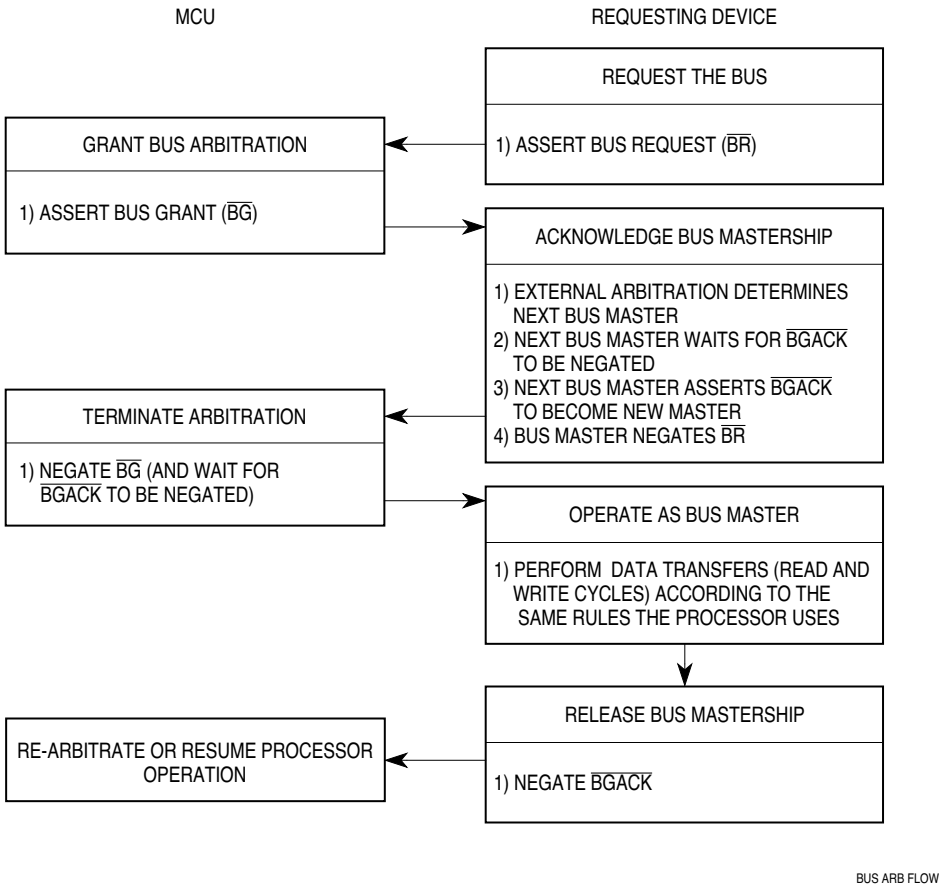


Figure 4-14 Bus Arbitration Flowchart for Single Request

State changes occur on the next rising edge of CLKOUT after the internal signal is valid. The $\overline{BG}$ signal transitions on the falling edge of the clock after a state is reached during which G changes. The bus control signals (controlled by T) are driven by the MCU immediately following a state change, when bus mastership is returned to the MCU. State 0, in which G and T are both negated, is the state of the bus arbiter while the MCU is bus master. Request R and acknowledge A keep the arbiter in state 0 as long as they are both negated.

4.5.6.1 Slave (Factory Test) Mode Arbitration

This mode is used for factory production testing of internal modules. It is not supported as a user operating mode. Slave mode is enabled by holding DATA11 low during reset. In slave mode, when $\overline{BG}$ is asserted, the MCU is slaved to an external master that has full access to all internal registers.

4.5.6.2 Show Cycles

The MCU normally performs internal data transfers without affecting the external bus, but it is possible to show these transfers during debugging. $\overline{AS}$ is not asserted externally during show cycles.

Show cycles are controlled by the SHEN field in the SIMCR (refer to **4.2.3 Show Internal Cycles**). This field is cleared by reset. When show cycles are disabled, the address bus, function codes, size, and read/write signals reflect internal bus activity, but $\overline{AS}$ and $\overline{DS}$ are not asserted externally and external data bus pins are in high-impedance state during internal accesses.

When show cycles are enabled, $\overline{DS}$ is asserted externally during internal cycles, and internal data is driven out on the external data bus. Because internal cycles normally continue to run when the external bus is granted, one SHEN encoding halts internal bus activity while there is an external master.

SIZ[1:0] signals reflect bus allocation during show cycles. Only the appropriate portion of the data bus is valid during the cycle. During a byte write to an internal address, the portion of the bus that represents the byte that is not written reflects internal bus conditions, and is indeterminate. During a byte write to an external address, the data multiplexer in the SIM causes the value of the byte that is written to be driven out on both bytes of the data bus.

4.6 Reset

Reset occurs when an active low logic level on the $\overline{RESET}$ pin is clocked into the SIM. The $\overline{RESET}$ input is synchronized to the system clock. If there is no clock when $\overline{RESET}$ is asserted, reset does not occur until the clock starts. Resets are clocked to allow completion of write cycles in progress at the time $\overline{RESET}$ is asserted.

Reset procedures handle system initialization and recovery from catastrophic failure. The MCU performs resets with a combination of hardware and software. The system integration module determines whether a reset is valid, asserts control signals, performs basic system configuration and boot ROM selection based on hardware mode-select inputs, then passes control to the CPU32.

4.6.1 Reset Exception Processing

The CPU32 processes resets as a type of asynchronous exception. An exception is an event that preempts normal processing, and can be caused by internal or external events. Exception processing makes the transition from normal instruction execution to execution of a routine that deals with an exception. Each exception has an assigned vector that points to an associated handler routine. These vectors are stored in the vector base register (VBR). The VBR contains the base address of a 1024-byte exception vector table, which consists of 256 exception vectors. The CPU32 uses vector numbers to calculate displacement into the table. Refer to **SECTION 5 CENTRAL PROCESSING UNIT** for more information concerning exceptions.

Reset is the highest-priority CPU32 exception. Unlike all other exceptions, a reset occurs at the end of a bus cycle, and not at an instruction boundary. Handling resets in this way prevents write cycles in progress at the time the reset signal is asserted from being corrupted. However, any processing in progress is aborted by the reset exception, and cannot be restarted. Only essential reset tasks are performed during exception processing. Other initialization tasks must be accomplished by the exception handler routine. **4.6.8 Reset Processing Summary** contains details of exception processing.

4.6.2 Reset Control Logic

SIM reset control logic determines the cause of a reset, synchronizes reset assertion if necessary to the completion of the current bus cycle, and asserts the appropriate reset lines. Reset control logic can drive four different internal signals.

1. EXTRST (external reset) drives the external reset pin.
2. CLKRST (clock reset) resets the clock module.
3. MSTRST (master reset) goes to all other internal circuits.
4. SYSRST (system reset) indicates to internal circuits that the CPU has executed a RESET instruction.

All resets are gated by CLKOUT. Resets are classified as synchronous or asynchronous. An asynchronous reset can occur on any CLKOUT edge. Reset sources that cause an asynchronous reset usually indicate a catastrophic failure; thus the reset control logic responds by asserting reset to the system immediately. (A system reset, however, caused by the CPU32 RESET instruction, is asynchronous but does not indicate any type of catastrophic failure).

Synchronous resets are timed (CLKOUT) to occur at the end of bus cycles. The internal bus monitor is automatically enabled for synchronous resets. When a bus cycle does not terminate normally, the bus monitor terminates it.

Refer to **Table 4-15** for a summary of reset sources.

Table 4-15 Reset Source Summary

Type	Source	Timing	Cause	Reset Lines Asserted by Controller		
External	External	Synch	External Signal	MSTRST	CLKRST	EXTRST
Power Up	EBI	Asynch	V_{DD}	MSTRST	CLKRST	EXTRST
Software Watchdog	Monitor	Asynch	Time Out	MSTRST	CLKRST	EXTRST
$\overline{\text{HALT}}$	Monitor	Asynch	Internal $\overline{\text{HALT}}$ Assertion (e.g. Double Bus Fault)	MSTRST	CLKRST	EXTRST
Loss of Clock	Clock	Synch	Loss of Reference	MSTRST	CLKRST	EXTRST
Test	Test	Synch	Test Mode	MSTRST	—	EXTRST
System	CPU32	Asynch	RESET Instruction	—	—	EXTRST

Internal single byte or aligned word writes are guaranteed valid for synchronous resets. External writes are also guaranteed to complete, provided the external configuration logic on the data bus is conditioned as shown in **Figure 4-13**.

4.6.3 Reset Mode Selection

The logic states of certain data bus pins during reset determine SIM operating configuration. In addition, the state of the MODCLK pin determines system clock source and the state of the $\overline{\text{BKPT}}$ pin determines what happens during subsequent breakpoint assertions. **Table 4-16** is a summary of reset mode selection options.

Table 4-16 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], $\overline{\text{AVEC}}$, $\overline{\text{DS}}$, $\overline{\text{AS}}$, $\overline{\text{SIZE}}$	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

4.6.3.1 Data Bus Mode Selection

All data lines have weak internal pull-up drivers. When pins are held high by the internal drivers, the MCU uses a default operating configuration. However, specific lines can be held low externally to achieve an alternate configuration.

NOTE

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. **Figure 4-15** shows a recommended method for conditioning the mode select signals.

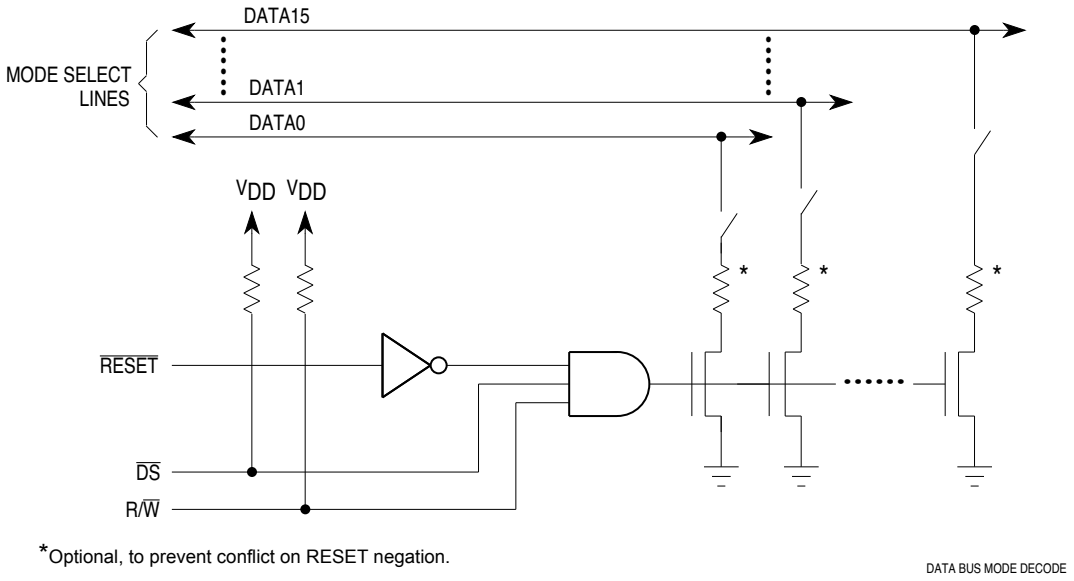


Figure 4-15 Data Bus Mode Select Conditioning

Data bus mode select current is specified in **APPENDIX A ELECTRICAL CHARACTERISTICS**. Do not confuse pin function with pin electrical state. Refer to **4.6.5 Pin State During Reset** for more information.

DATA0 determines the function of the boot ROM chip-select signal ($\overline{\text{CSBOOT}}$). Unlike other chip-select signals, $\overline{\text{CSBOOT}}$ is active at the release of reset. During reset exception processing, the MCU fetches initialization vectors beginning at address $000000 in supervisor program space. An external memory device containing vectors located at these addresses can be enabled by $\overline{\text{CSBOOT}}$ after a reset. The logic level of DATA0 during reset selects boot ROM port size for dynamic bus allocation. When DATA0 is held low, port size is eight bits; when DATA0 is held high, either by the weak internal pull-up driver or by an external pull-up, port size is 16 bits. Refer to **4.8.4 Chip-Select Reset Operation** for more information.

DATA1 and DATA2 determine the functions of $\overline{\text{CS}}$[2:0] and $\overline{\text{CS}}$[5:3], respectively. DATA[7:3] determine the functions of an associated chip select and all lower-numbered chip-selects down through $\overline{\text{CS6}}$. For example, if DATA5 is pulled low during reset, $\overline{\text{CS}}$[8:6] are assigned alternate function as ADDR[21:19], and $\overline{\text{CS}}$[10:9] remain chip-selects. Refer to **4.8.4 Chip-Select Reset Operation** for more information.

DATA8 determines the function of the $\overline{\text{DSACK}}$[1:0], $\overline{\text{AVEC}}$, $\overline{\text{DS}}$, $\overline{\text{AS}}$, and SIZE pins. If DATA8 is held low during reset, these pins are assigned to I/O port E.

DATA9 determines the function of interrupt request pins $\overline{\text{IRQ}}$[7:0] and the clock mode select pin (MODCLK). When DATA9 is held low during reset, these pins are assigned to I/O port F.

DATA11 determines whether the SIM operates in test mode out of reset. This capability is used for factory testing of the MCU.

4.6.3.2 Clock Mode Selection

The state of the clock mode (MODCLK) pin during reset determines what clock source the MCU uses. When MODCLK is held high during reset, the clock signal is generated from a reference frequency. When MODCLK is held low during reset, the clock synthesizer is disabled, and an external system clock signal must be applied. Refer to **4.3 System Clock** for more information.

NOTE

The MODCLK pin can also be used as parallel I/O pin PF0. To prevent inadvertent clock mode selection by logic connected to port F, use an active device to drive MODCLK during reset.

4.6.3.3 Breakpoint Mode Selection

The MCU uses internal and external breakpoint ($\overline{\text{BKPT}}$) signals. During reset exception processing, at the release of the $\overline{\text{RESET}}$ signal, the CPU32 samples these signals to determine how to handle breakpoints.

If either $\overline{\text{BKPT}}$ signal is at logic level zero when sampled, an internal BDM flag is set, and the CPU32 enters background debugging mode whenever either $\overline{\text{BKPT}}$ input is subsequently asserted.

If both $\overline{\text{BKPT}}$ inputs are at logic level one when sampled, breakpoint exception processing begins whenever either $\overline{\text{BKPT}}$ signal is subsequently asserted.

Refer to **SECTION 5 CENTRAL PROCESSING UNIT** for more information on background debugging mode and exceptions. Refer to **4.5.4 CPU Space Cycles** for information concerning breakpoint acknowledge bus cycles.

4.6.4 MCU Module Pin Function During Reset

Usually, module pins 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. **Table 4-17** is a summary of module pin function out of reset. Refer to **APPENDIX D REGISTER SUMMARY** for register function and reset state.

Table 4-17 Module Pin Functions

Module	Pin Mnemonic	Function
CPU32	DSI/$\overline{\text{IFETCH}}$	DSI/$\overline{\text{IFETCH}}$
	DSO/$\overline{\text{IPIPE}}$	DSO/$\overline{\text{IPIPE}}$
	$\overline{\text{BKPT}}$/DSCLK	$\overline{\text{BKPT}}$/DSCLK
GPT	PGP7/IC4/OC5	Discrete Input
	PGP[6:3]/OC[4:1]	Discrete Input
	PGP[2:0]/IC[3:1]	Discrete Input
	PAI	Discrete Input
	PCLK	Discrete Input
	PWMA, PWMB	Discrete Output
QSM	PQS7/TXD	Discrete Input
	PQS[6:4]/PCS[3:1]	Discrete Input
	PQS3/PCS0/$\overline{\text{SS}}$	Discrete Input
	PQS2/SCK	Discrete Input
	PQS1/MOSI	Discrete Input
	PQS0/MISO	Discrete Input
	RXD	RXD

4.6.5 Pin State During Reset

It is important to keep the distinction between pin function and pin electrical state clear. Although control register values and mode select inputs determine pin function, a pin driver can be active, inactive or in high-impedance state while reset occurs. During power-up reset, pin state is subject to the constraints discussed in **4.6.7 Power-On Reset**.

NOTE

Pins that are not used should either be configured as outputs, or (if configured as inputs) pulled to the appropriate inactive state. This decreases additional I_{DD} caused by digital inputs floating near mid-supply level.

4.6.5.1 Reset States of SIM Pins

Generally, while $\overline{\text{RESET}}$ is asserted, SIM 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 during reset become active high-impedance loads after $\overline{\text{RESET}}$ is released. 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 4-18** is a summary of SIM pin states during reset.

Table 4-18 SIM Pin Reset States

Mnemonic	State While RESET Asserted	Pin State After RESET Released			
		Pin Function	Pin State	Pin Function	Pin State
CS10/ADDR23	1	CS10	1	ADDR23	Unknown
CS[9:6]/ADDR[22:19]/PC[6:3]	1	CS[9:6]	1	ADDR[22:19]	Unknown
ADDR[18:0]	High-Z Output	ADDR[18:0]	Unknown	ADDR[18:0]	Unknown
AS/PE5	High-Z Output	AS	Output	PE5	Input
AVEC/PE2	Disabled	AVEC	Input	PE2	Input
BERR	Disabled	BERR	Input	BERR	Input
CSM/BG	1	CSM	1	BG	1
CSE/BGACK	1	CSE	1	BGACK	Input
CS0/BR	1	CS0	1	BR	Input
CLKOUT	Output	CLKOUT	Output	CLKOUT	Output
CSBOOT	1	CSBOOT	0	CSBOOT	0
DATA[15:0]	Mode Select	DATA[15:0]	Input	DATA[15:0]	Input
DS/PE4	Disabled	DS	Output	PE4	Input
DSACK0/PE0	Disabled	DSACK0	Input	PE0	Input
DSACK1/PE1	Disabled	DSACK1	Input	PE1	Input
CS5/FC2/PC2	1	CS5	1	FC2	Unknown
FC1/PC1	1	FC1	1	FC1	Unknown
CS3/FC0/PC0	1	CS3	1	FC0	Unknown
HALT	Disabled	HALT	Input	HALT	Input
IRQ[7:1]/PF[7:1]	Disabled	IRQ[7:1]	Input	PF[7:1]	Input
MODCLK/PF0	Mode Select	MODCLK	Input	PF0	Input
R/W	Disabled	R/W	Output	R/W	Output
RESET	Asserted	RESET	Input	RESET	Input
RMC	Disabled	RMC	Output	PE3	Input
SIZ[1:0]/PE[7:6]	Disabled	SIZ[1:0]	Unknown	PE[7:6]	Input
TSC	Mode Select	TSC	Input	TSC	Input

4.6.5.2 Reset States of Pins Assigned to Other MCU Modules

As a rule, module pins that are assigned to general-purpose I/O ports go to active high-impedance state following reset. Other pin states are determined by individual module control register settings. Refer to sections concerning modules for details. However, during power-up reset, module port pins may be in an indeterminate state for a short period. Refer to **4.6.7 Power-On Reset** for more information.

4.6.6 Reset Timing

The RESET input must be asserted for a specified minimum period for reset to occur. External 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 RESET is asserted, SIM pins are either in an inactive, high impedance state or are driven to their inactive states.

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

If an internal source asserts a 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 ten 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.

4.6.7 Power-On Reset

When the SIM clock synthesizer is used to generate system clocks, 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_{DDSYN} for the MCU to operate. The following discussion assumes that V_{DDSYN} is applied before and during reset, which minimizes crystal start-up time. When V_{DDSYN} is applied at power-on, start-up time is affected by specific crystal parameters and by oscillator circuit design. V_{DD} ramp-up time also affects pin state during reset. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for voltage and timing specifications.

During power-on reset, an internal circuit in the SIM drives the IMB internal (MSTRST) and external (EXTRST) reset lines. The circuit releases MSTRST as V_{DD} ramps up to the minimum specified value, and SIM pins are initialized as shown in **Table 4-19**. As V_{DD} reaches 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 SIM clock synthesizer provides clock signals to the other MCU modules. After the clock is running and MSTRST is asserted for at least four clock cycles, these modules reset. V_{DD} ramp time and VCO frequency ramp time determine how long the 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 external pull-up resistors, external logic on input/output or output-only pins during this time must condition the lines. Active drivers require high-impedance buffers or isolation resistors to prevent conflict.

Figure 4-16 is a timing diagram of power-up reset. It shows the relationships between $\overline{\text{RESET}}$, V_{DD}, and bus signals.

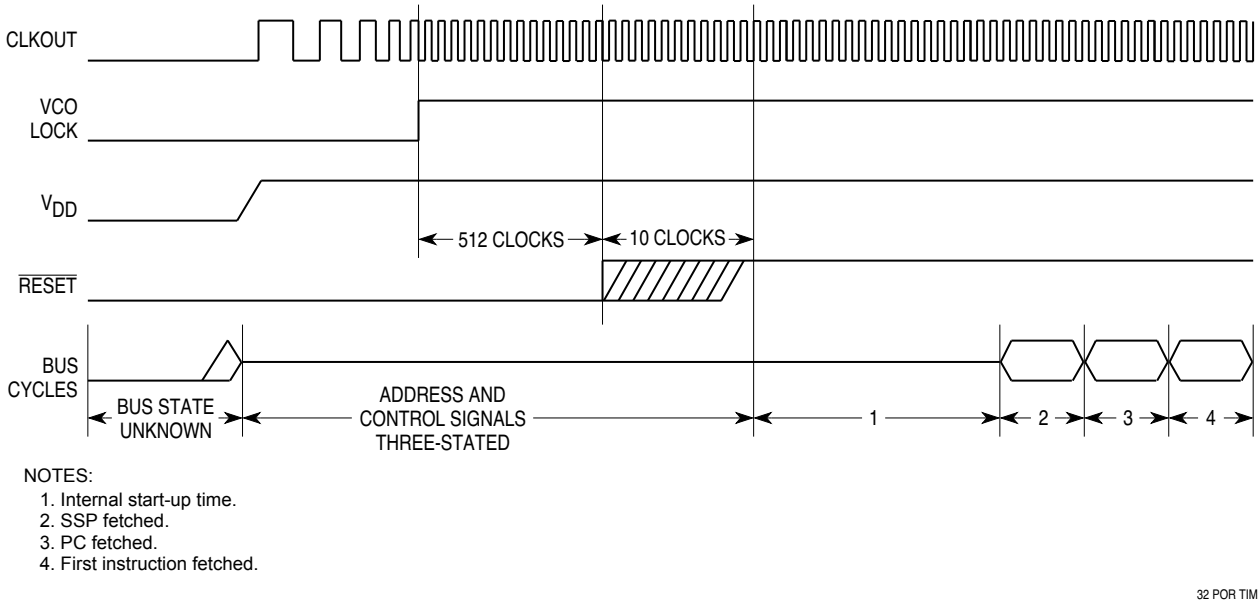


Figure 4-16 Power-On Reset

4.6.8 Reset Processing Summary

To prevent write cycles in progress from being corrupted, a reset is recognized at the end of a bus cycle, and not at an instruction boundary. Any processing in progress at the time a reset occurs is aborted. After SIM reset control logic has synchronized an internal or external reset request, it asserts the MSTRST signal.

The following events take place when MSTRST is asserted.

A. Instruction execution is aborted.
B. The status register is initialized.
   1. The T0 and T1 bits are cleared to disable tracing.
   2. The S bit is set to establish supervisor privilege level.
   3. The interrupt priority mask is set to $7, disabling all interrupts below priority 7.
C. The vector base register is initialized to $000000.

The following events take place when MSTRST is negated after assertion.

A. The CPU32 samples the $\overline{\text{BKPT}}$ input.
B. The CPU32 fetches the reset vector:
   1. The first long word of the vector is loaded into the interrupt stack pointer.
   2. The second long word of the vector is loaded into the program counter.

   Vectors can be fetched from internal RAM or from external ROM enabled by the $\overline{\text{CSBOOT}}$ signal.

C. The CPU32 fetches and begins decoding the first instruction to be executed.

4.6.9 Reset Status Register

The reset status register (RSR) contains a bit for each reset source in the MCU. When a reset occurs, a bit corresponding to the reset type is set. When multiple causes of reset occur at the same time, more than one bit in RSR may be set. The reset status register is updated by the reset control logic when the $\overline{\text{RESET}}$ signal is released. Refer to **APPENDIX D REGISTER SUMMARY**.

4.7 Interrupts

Interrupt recognition and servicing involve complex interaction between the system integration module, the central processing unit, and a device or module requesting interrupt service. This discussion provides an overview of the entire interrupt process. Chip-select logic can also be used to respond to interrupt requests. Refer to **4.8 Chip Selects** for more information.

4.7.1 Interrupt Exception Processing

The CPU32 processes resets as a type of asynchronous exception. An exception is an event that preempts normal processing. Each exception has an assigned vector in an exception vector table that points to an associated handler routine. The CPU uses vector numbers to calculate displacement into the table. During exception processing, the CPU fetches the appropriate vector and executes the exception handler routine to which the vector points.

Out of reset, the exception vector table is located beginning at address $000000. This value can be changed by programming the vector base register (VBR) with a new value, and multiple vector tables can be used. Refer to **SECTION 5 CENTRAL PROCESSING UNIT** for more information concerning exceptions.

4.7.2 Interrupt Priority and Recognition

The CPU32 provides eight levels of interrupt priority. All interrupts with priorities less than seven can be masked by the interrupt priority (IP) field in status register.

There are seven interrupt request signals ($\overline{\text{IRQ}}$[7:1]). These signals are used internally on the IMB, and 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 SIM. Each of the interrupt request signals corresponds to an interrupt priority level. $\overline{\text{IRQ1}}$ has the lowest priority and $\overline{\text{IRQ7}}$ 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 CPU32 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. $\overline{\text{IRQ7}}$, 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.

$\overline{\text{IRQ7}}$ is transition-sensitive as well as level-sensitive: a level-7 interrupt is not detected unless a falling edge transition is detected on the $\overline{\text{IRQ7}}$ line. This prevents redundant servicing and stack overflow. A nonmaskable interrupt is generated each time $\overline{\text{IRQ7}}$ is asserted as well as each time the priority mask changes from %111 to a lower number while $\overline{\text{IRQ7}}$ 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 CPU32 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 CPU32 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 CPU32 recognizes the higher-level request.

4.7.3 Interrupt Acknowledge and Arbitration

When the CPU32 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 CPU32 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 SIM, 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 nonzero value, the CPU32 interprets multiple vector numbers at the same time, with unpredictable consequences.

Because the EBI manages external interrupt requests, the SIM IARB value is used for arbitration between internal and external interrupt requests. The reset value of IARB for the SIM 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 SIM 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 ($\overline{\text{DSACK}}$) termination signals, or it must assert the autovector ($\overline{\text{AVEC}}$) request signal. If the device does not respond in time, the EBI bus monitor asserts the bus error signal $\overline{\text{BERR}}$, and a spurious interrupt exception is taken.

Chip-select logic can also be used to generate internal $\overline{\text{AVEC}}$ or $\overline{\text{DSACK}}$ signals in response to interrupt requests from external devices (refer to **4.8.3 Using Chip-Select Signals for Interrupt Acknowledge**). 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 $\overline{\text{AVEC}}$ or $\overline{\text{DSACK}}$ 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 PIRQ field in the periodic interrupt control register (PICR) determines PIT priority level. A PIRQ value of %000 means that PIT interrupts are inactive. By hardware convention, when the CPU32 receives simultaneous interrupt requests of the same level from more than one SIM source (including external devices), the periodic interrupt timer is given the highest priority, followed by the $\overline{\text{IRQ}}$ pins.

4.7.4 Interrupt Processing Summary

A summary of the entire 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.

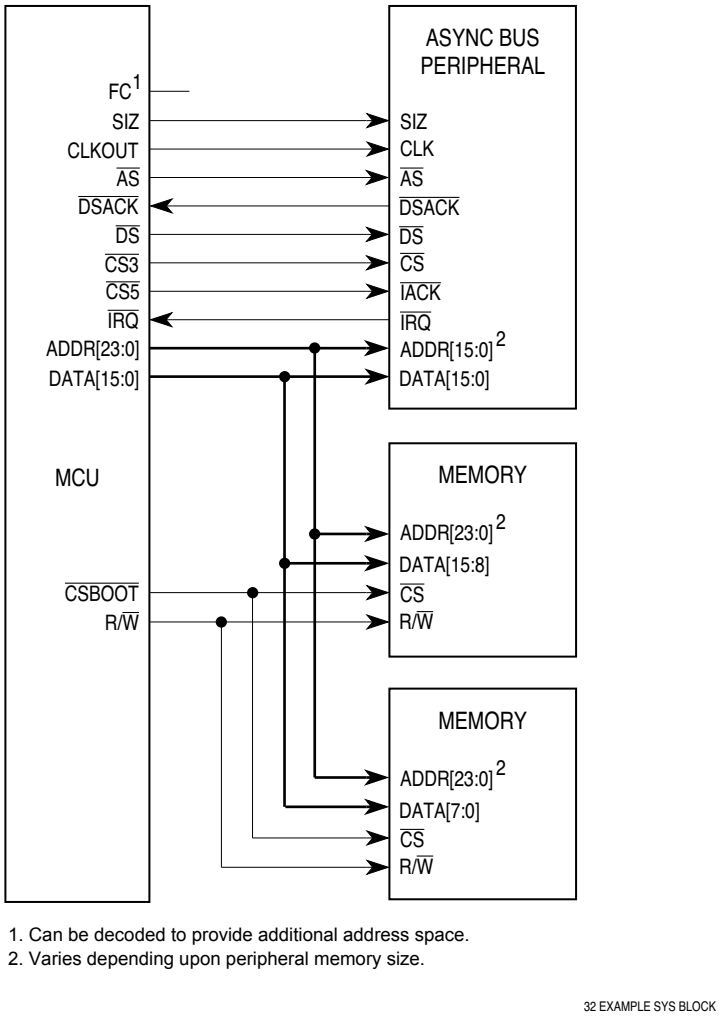
B. The processor state is stacked. The S bit in the status register is set, establishing supervisor access level, and bits T1 and T0 are cleared, disabling tracing.
C. The interrupt acknowledge cycle begins:
   1. FC[2:0] are driven to %111 (CPU space) encoding.
   2. The address bus is driven as follows: 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 in 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 CPU32 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.

4.7.5 Interrupt Acknowledge Bus Cycles

Interrupt acknowledge bus cycles are CPU32 space cycles that are generated during exception processing. For further information about the types of interrupt acknowledge bus cycles determined by $\overline{\text{AVEC}}$ or $\overline{\text{DSACK}}$, refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** and the *SIM Reference Manual* (SIMRM/AD).

4.8 Chip Selects

Typical microcontrollers require additional hardware to provide external chip-select and address decode signals. The MCU includes 12 programmable chip-select circuits that can provide 2- to 20-clock-cycle access to external memory and peripherals. Address block sizes of two Kbytes to one Mbyte can be selected. **Figure 4-17** is a diagram of a basic system that uses chip selects.



1. Can be decoded to provide additional address space.
2. Varies depending upon peripheral memory size.

32 EXAMPLE SYS BLOCK

Figure 4-17 Basic MCU System

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 DSACK and AVEC signals internally. Each signal can also be synchronized with the ECLK signal available on ADDR23.

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 are configured for operation out of reset. However, all chip-select signals except CSBOOT are disabled, and cannot be asserted until the BYTE field

in the corresponding option register is programmed to a nonzero value, selecting a transfer size. The chip-select option must not be written until a base address has been written to a proper base address register. $\overline{\text{CSBOOT}}$ is automatically asserted out of reset. Alternate functions for chip-select pins are enabled if appropriate data bus pins are held low at the release of the reset signal (refer to **4.6.3.1 Data Bus Mode Selection** for more information). **Figure 4-18** is a functional diagram of a single chip-select circuit.

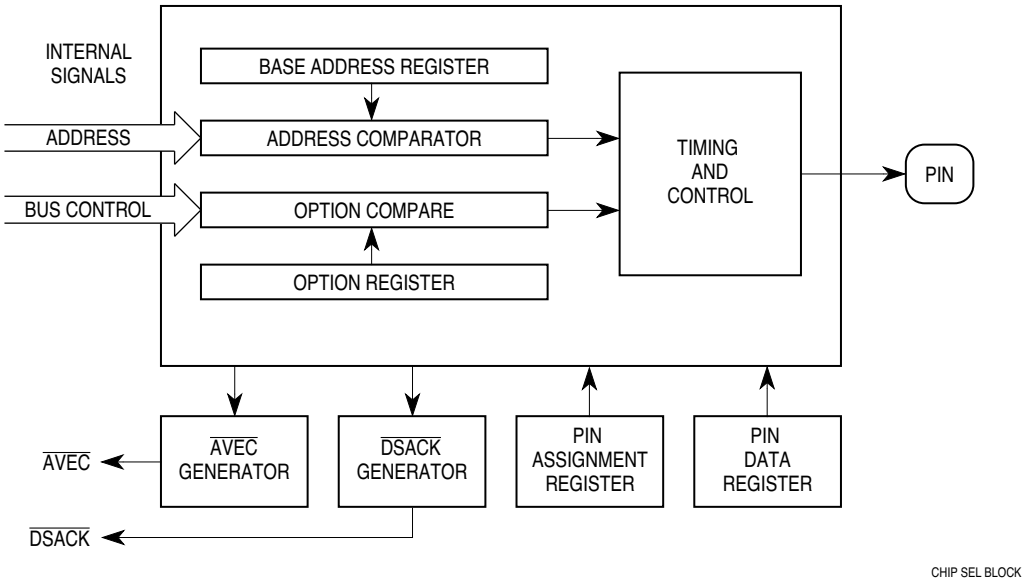


Figure 4-18 Chip-Select Circuit Block Diagram

4.8.1 Chip-Select Registers

Each chip-select pin can have one or more functions. Chip-select pin assignment registers (CSPAR[0:1]) determine functions of the pins. Pin assignment registers also determine port size (8- or 16-bit) for dynamic bus allocation. A pin data register (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 register (CSBAR[0:10], CSBARBT). Address blocks for separate chip-select functions can overlap.

Chip select option registers (CSOR[0:10], CSORBT) 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. A set of special chip-select functions and registers (CSORBT, CSBARBT) is provided to support bootstrap operation.

Comprehensive address maps and register diagrams are provided in **APPENDIX D REGISTER SUMMARY**.

4.8.1.1 Chip-Select Pin Assignment Registers

The pin assignment registers contain twelve 2-bit fields ($\overline{CS}$[10:0] and $\overline{CSBOOT}$) that determine the functions of the chip-select pins. Each pin has two or three possible functions, as shown in **Table 4-19**.

Table 4-19 Chip-Select Pin Functions

16-Bit Chip Select	8-Bit Chip Select	Alternate Function	Discrete Output
$\overline{CSBOOT}$	$\overline{CSBOOT}$	$\overline{CSBOOT}$	—
$\overline{CS0}$	$\overline{CS0}$	$\overline{BR}$	—
$\overline{CS1}$	$\overline{CS1}$	$\overline{BG}$	—
$\overline{CS2}$	$\overline{CS2}$	$\overline{BGACK}$	—
$\overline{CS3}$	$\overline{CS3}$	FC0	PC0
$\overline{CS4}$	$\overline{CS4}$	FC1	PC1
$\overline{CS5}$	$\overline{CS5}$	FC2	PC2
$\overline{CS6}$	$\overline{CS6}$	ADDR19	PC3
$\overline{CS7}$	$\overline{CS7}$	ADDR20	PC4
$\overline{CS8}$	$\overline{CS8}$	ADDR21	PC5
$\overline{CS9}$	$\overline{CS9}$	ADDR22	PC6
$\overline{CS10}$	$\overline{CS10}$	ADDR23	ECLK

Table 4-20 shows pin assignment field encoding. Pins that have no discrete output function do not use the %00 encoding.

Table 4-20 Pin Assignment Field Encoding

Bit Field	Description
00	Discrete Output
01	Alternate Function
10	Chip Select (8-Bit Port)
11	Chip Select (16-Bit Port)

Port size determines the way in which bus transfers to an external address 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 **4.8.1.3 Chip-Select 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. (Refer to **4.6.3.1 Data Bus Mode Selection** for more information.) 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. There are twelve chip-select functions and only eight associated data bus pins. There is not a one-to-one correspondence. Refer to **4.8.4 Chip-Select Reset Operation** for more detailed information.

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 $\overline{\text{CSBOOT}}$, $\overline{\text{BR}}$, $\overline{\text{BG}}$, or $\overline{\text{BGACK}}$. 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 $\overline{\text{DSACK}}$ or $\overline{\text{AVEC}}$ internally on an address and control signal match.

4.8.1.2 Chip-Select 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. Block addresses for different chip selects can overlap.

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. **Table 4-21** shows BLKSZ encoding.

Table 4-21 Block Size Encoding

BLKSZ[2:0]	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]

The chip-select 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 block size. Base address register diagrams show how base register bits correspond to address lines.

After reset, the MCU fetches the initialization routine from the address contained in the reset vector, located beginning at address $000000 of program space. To support bootstrap operation from reset, the base address field in chip-select base address register boot (CSBARBT) has a reset value of all zeros. A memory device containing the reset vector and initialization routine can be automatically enabled by $\overline{\text{CSBOOT}}$ after a reset. The block size field in CSBARBT has a reset value of 512 Kbytes. Refer to **4.8.4 Chip-Select Reset Operation** for more information.

4.8.1.3 Chip-Select Option Registers

Option register fields determine timing of and conditions for assertion of chip-select signals. To assert a chip-select signal, and to provide $\overline{\text{DSACK}}$ or autovector support, other constraints set by fields in the option register and in the base address register must also be satisfied. **Table 4-22** is a summary of option register functions.

Table 4-22 Option Register Function Summary

MODE	BYTE	R/$\overline{W}$	STRB	DSACK	SPACE	IPL	AVEC
0 = ASYNC*	00 = Disable	00 = Rsvd	0 = $\overline{AS}$	0000 = 0 WAIT	00 = CPU SP	000 = All*	0 = Off*
1 = SYNC	01 = Lower	01 = Read	1 = $\overline{DS}$	0001 = 1 WAIT	01 = User SP	001 = Priority 1	1 = On
	10 = Upper	10 = Write		0010 = 2 WAIT	10 = Supv SP	010 = Priority 2	
	*11 = Both	11 = Both		0011 = 3 WAIT	11 = S/U SP*	011 = Priority 3	
				0100 = 4 WAIT		100 = Priority 4	
				0101 = 5 WAIT		101 = Priority 5	
				0110 = 6 WAIT		110 = Priority 6	
				0111 = 7 WAIT		111 = Priority 7	
				1000 = 8 WAIT			
				1001 = 9 WAIT			
				1010 = 10 WAIT			
				1011 = 11 WAIT			
				1100 = 12 WAIT			
				1101 = 13 WAIT			
				1110 = F term			
				1111 = External			

*Use this value when function is not required for chip-select operation.

The **MODE** bit determines whether chip-select assertion simulates an asynchronous bus cycle, or is synchronized to the M6800-type bus clock signal (ECLK) available on ADDR23 (refer to **4.3 System Clock** for more information on ECLK).

The **BYTE** field controls bus allocation for chip-select transfers. Port size, set when a chip select is enabled by a pin assignment register, affects signal assertion. When an 8-bit port is assigned, any BYTE field value other than %00 enables the chip select signal. When a 16-bit port is assigned, however, BYTE field value determines when the chip select is enabled. The BYTE fields for $\overline{CS}$[10:0] are cleared during reset. However, both bits in the boot ROM option register (CSORBT) BYTE field are set (%11) when the reset signal is released.

The **R/W** field causes a chip-select signal to be asserted only for a read, only for a write, or for both read and write. Use this field in conjunction with the STRB bit to generate asynchronous control signals for external devices.

The **STRB** bit controls the timing of a chip-select assertion in asynchronous mode. Selecting address strobe causes a chip-select signal to be asserted synchronized with the address strobe. Selecting data strobe causes a chip-select signal to be asserted synchronized with the data strobe. This bit has no effect in synchronous mode.

The **DSACK** field specifies the source of data strobe acknowledge signals used in asynchronous mode. It also allows the user to optimize bus speed in a particular application by controlling the number of wait states that are inserted.

The **SPACE** field determines the address space in which a chip select is asserted. An access must have the space type represented by SPACE encoding in order for a chip-select signal to be asserted.

The **IPL** field contains an interrupt priority mask that is used when chip-select logic is set to trigger on external interrupt acknowledge cycles. When the SPACE field is set

to %00 (CPU space), interrupt priority (ADDR[3:1]) is compared to IPL value. If the values are the same, and other option register constraints are satisfied, a chip select signal is asserted. This field only affects the response of chip selects and does not affect interrupt recognition by the CPU. Encoding %000 causes a chip-select signal to be asserted regardless of interrupt acknowledge cycle priority, provided all other constraints are met.

The **AVEC** bit selects one of two methods of acquiring an interrupt vector during an external interrupt acknowledge cycle. The internal autovector signal is generated only in response to interrupt requests from the SIM $\overline{IRQ}$ pins.

4.8.1.4 PORTC Data Register

The PORTC data register latches data for PORTC pins programmed as discrete outputs. When a pin is assigned as a discrete output, the value in this register appears at the output. PC[6:0] correspond to $\overline{CS}$[9:3]. Bit 7 is not used. Writing to this bit has no effect, and it always reads zero.

4.8.2 Chip-Select Operation

When the MCU makes an access, enabled chip-select circuits compare the following items:

1. Function codes to SPACE fields, and to the IPL field if the SPACE field encoding is not for CPU32 space.
2. Appropriate ADDR bits to base address fields.
3. Read/write status to R/$\overline{W}$ fields.
4. ADDR0 and/or SIZ bits to the BYTE field (16-bit ports only).
5. Priority of the interrupt being acknowledged (ADDR[3:1]) to IPL fields (when the access is an interrupt acknowledge cycle).

When a match occurs, the chip-select signal is asserted. Assertion occurs at the same time as $\overline{AS}$ or $\overline{DS}$ assertion in asynchronous mode. Assertion is synchronized with ECLK in synchronous mode. In asynchronous mode, the value of the $\overline{DSACK}$ field determines whether $\overline{DSACK}$ is generated internally.$\overline{DSACK}$ also determines the number of wait states inserted before internal $\overline{DSACK}$ assertion.

The speed of an external device determines whether internal wait states are needed. Normally, wait states are inserted into the bus cycle during S3 until a peripheral asserts $\overline{DSACK}$. If a peripheral does not generate $\overline{DSACK}$, internal $\overline{DSACK}$ generation must be selected and a predetermined number of wait states can be programmed into the chip-select option register.

Refer to the *SIM Reference Manual* (SIMRM/AD) for further information.

4.8.3 Using Chip-Select Signals for Interrupt Acknowledge

Ordinary I/O bus cycles use supervisor space access, but interrupt acknowledge bus cycles use CPU space access. Refer to **4.5.4 CPU Space Cycles** and **4.7 Interrupts** for more information. There are no differences in flow for chip selects in each type of space, but base and option registers must be properly programmed for each type of external bus cycle.

During a CPU space cycle, bits [15:3] of the appropriate base register must be configured to match ADDR[23:11], as the address is compared to an address generated by the CPU.

Figure 4-19 shows CPU space encoding for an interrupt acknowledge cycle. FC[2:0] are set to %111, designating CPU space access. ADDR[3:1] indicate interrupt priority, and the space type field (ADDR[19:16]) is set to %1111, the interrupt acknowledge code. The rest of the address lines are set to one.

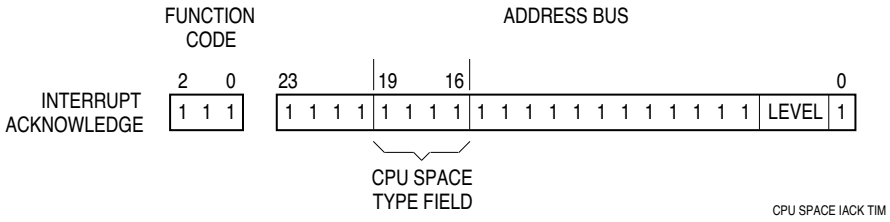


Figure 4-19 CPU Space Encoding for Interrupt Acknowledge

Because address match logic functions only after the EBI transfers an interrupt acknowledge cycle to the external address bus following IARB contention, chip-select logic generates $\overline{\text{AVEC}}$ or $\overline{\text{DSACK}}$ signals only in response to interrupt requests from external $\overline{\text{IRQ}}$ pins. If an internal module makes an interrupt request of a certain priority, and the chip-select base address and option registers are programmed to generate $\overline{\text{AVEC}}$ or $\overline{\text{DSACK}}$ 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 an internal $\overline{\text{DSACK}}$ signal to terminate the cycle.

Perform the following operations before using a chip select to generate an interrupt acknowledge signal.

1. Program the base address field to all ones.
2. Program block size to no more than 64 Kbytes, so that the address comparator checks ADDR[19:16] against the corresponding bits in the base address register. (The CPU32 places the CPU32 space type on ADDR[19:16].)
3. Set the R/$\overline{\text{W}}$ field to read only. An interrupt acknowledge cycle is performed as a read cycle.
4. Set the BYTE field to lower byte when using a 16-bit port, as the external vector for a 16-bit port is fetched from the lower byte. Set the BYTE field to upper byte when using an 8-bit port.

If an interrupting device does not provide a vector number, an autovector acknowledge must be generated. Asserting $\overline{\text{AVEC}}$, either by asserting the $\overline{\text{AVEC}}$ pin or by generating $\overline{\text{AVEC}}$ internally using the chip-select option register, terminates the bus cycle.

4.8.4 Chip-Select Reset Operation

The least significant bits of each of the 2-bit $\overline{\text{CS}}$[10:0] pin assignment fields in CSPAR0 and CSPAR1 each have a reset value of one. The reset values of the most significant bits of each field are determined by the states of DATA[7:1] during reset. There are

weak internal pull-up drivers for each of the data lines, so that chip-select operation will be selected by default out of reset. However, the internal pull-up drivers can be overcome by bus loading effects — to insure a particular configuration out of reset, use an active device to put the data lines in a known state during reset. The base address fields in chip-select base address registers CSBAR[0:10] and chip select option registers CSOR[0:10] have the reset values shown in **Table 4-23**. The BYTE fields of CSOR[0:10] have a reset value of “disable”, so that a chip-select signal cannot be asserted until the base and option registers are initialized.

Table 4-23 Chip Select Base and Option Register Reset Values

Fields	Reset Values
Base Address	$000000
Block Size	2 Kbyte
Async/Sync Mode	Asynchronous Mode
Upper/Lower Byte	Disabled
Read/Write	Reserved
$\overline{AS}$/$\overline{DS}$	$\overline{AS}$
$\overline{DSACK}$	No Wait States
Address Space	CPU Space
IPL	Any Level
Autovector	External Interrupt Vector

Following reset, the MCU fetches initial stack pointer and program counter values from the exception vector table, beginning at $000000 in supervisor program space. The $\overline{CSBOOT}$ chip-select signal is used to select an external boot ROM mapped to a base address of $000000. In order to do this, the reset values of the fields that control $\overline{CSBOOT}$ must be different from those of other chip select signals.

The MSB of the $\overline{CSBOOT}$ field in CSPAR0 has a reset value of one, so that chip-select function is selected by default out of reset. The BYTE field in option register CSORBT has a reset value of “both bytes” so that the select signal is enabled out of reset. The LSB value of the $\overline{CSBOOT}$ field, determined by the logic level of DATA0 during reset, selects boot ROM port size. When DATA0 is held low during reset, port size is eight bits. When DATA0 is held high during reset, port size is 16 bits. DATA0 has a weak internal pull-up driver, so that a 16-bit port will be selected by default out of reset. However, the internal pull-up driver can be overcome by bus loading effects —to insure a particular configuration out of reset, use an active device to put DATA0 in a known state during reset.

The base address field in chip-select base address register boot (CSBARBT) has a reset value of all zeros, so that when the initial access to address $000000 is made, an address match occurs, and the $\overline{CSBOOT}$ signal is asserted. The block size field in CSBARBT has a reset value of 1 Mbyte. **Table 4-24** shows $\overline{CSBOOT}$ reset values.

Table 4-24 $\overline{\text{CSBOOT}}$ Base and Option Register Reset Values

Fields	Reset Values
Base Address	$000000
Block Size	1 Mbyte
Async/Sync Mode	Asynchronous Mode
Upper/Lower Byte	Both Bytes
Read/Write	Read/Write
$\overline{\text{AS}}$/$\overline{\text{DS}}$	$\overline{\text{AS}}$
$\overline{\text{DSACK}}$	13 Wait States
Address Space	Supervisor/User Space
IPL	Any Level
Autovector	Interrupt Vector Externally

4.9 Parallel Input/Output Ports

Fifteen SIM pins can be configured for general-purpose discrete input and output. Although these pins are organized into two ports, port E and port F, function assignment is by individual pin. Pin assignment registers, data direction registers, and data registers are used to implement discrete I/O.

4.9.1 Pin Assignment Registers

Bits in the port E and port F pin assignment registers (PEPAR and PFPAR) control the functions of the pins in each port. Any bit set to one defines the corresponding pin as a bus control signal. Any bit cleared to zero defines the corresponding pin as an I/O pin.

4.9.2 Data Direction Registers

Bits in the port E and port F data direction registers (DDRE and DDRF) control the direction of the pin drivers when the pins are configured as I/O. Any bit in a register set to one configures the corresponding pin as an output. Any bit in a register cleared to zero configures the corresponding pin as an input. These registers can be read or written at any time. Writes have no effect.

4.9.3 Data Registers

A write to the port E and port F data registers (PORTE and PORTF) is stored in an internal data latch, and if any pin in the corresponding port is configured as an output, the value stored for that bit is driven out on the pin. A read of a 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. Both data registers can be accessed in two locations. Registers can be read or written at any time.

4.10 Factory Test

The test submodule supports scan-based testing of the various MCU modules. It is integrated into the SIM to support production test. Test submodule registers are intended for Freescale use only. Register names and addresses are provided in **APPENDIX D REGISTER SUMMARY** to show the user that these addresses are occupied. The QUOT pin is also used for factory test.

SECTION 5 CENTRAL PROCESSING UNIT

The CPU32, the instruction processing module of the M68300 family, is based on the industry-standard MC68000 processor. It has many features of the MC68010 and MC68020, as well as unique features suited for high-performance controller applications. This section is an overview of the CPU32. For detailed information concerning CPU operation, refer to the *CPU32 Reference Manual* (CPU32RM/AD).

5.1 General

Ease of programming is an important consideration in using a microcontroller. The CPU32 instruction format reflects a philosophy emphasizing register-memory interaction. There are eight multifunction data registers and seven general-purpose addressing registers.

All data resources are available to all operations requiring those resources. The data registers readily support 8-bit (byte), 16-bit (word), and 32-bit (long-word) operand lengths for all operations. Word and long-word operations support address manipulation. Although the program counter (PC) and stack pointers (SP) are special-purpose registers, they are also available for most data addressing activities. Ease of program checking and diagnosis is further enhanced by trace and trap capabilities at the instruction level.

A block diagram of the CPU32 is shown in **Figure 5-1**. The major blocks operate in a highly independent fashion that maximizes concurrence of operation while managing the essential synchronization of instruction execution and bus operation. The bus controller loads instructions from the data bus into the decode unit. The sequencer and control unit provide overall chip control, managing the internal buses, registers, and functions of the execution unit.

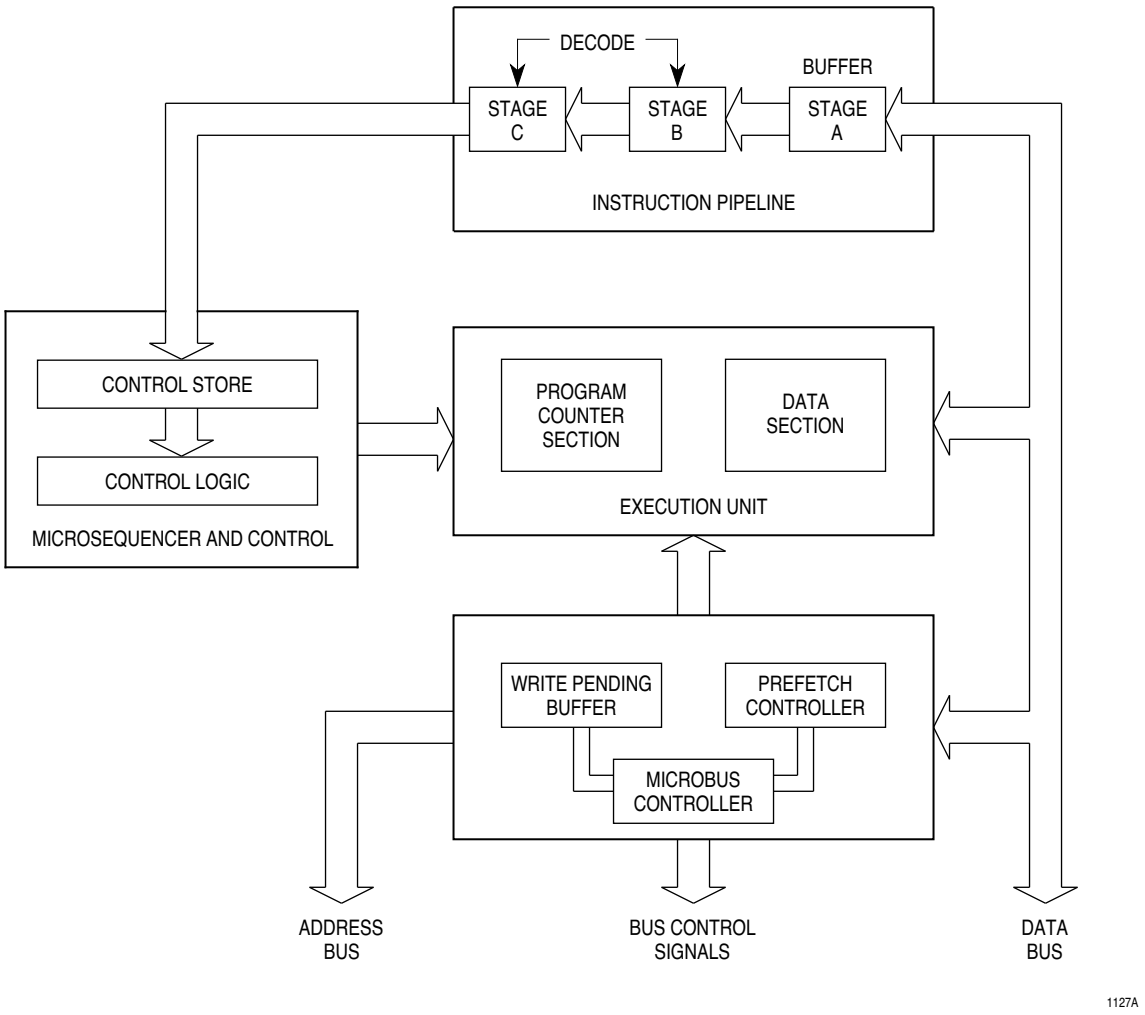


Figure 5-1 CPU32 Block Diagram

5.2 CPU32 Registers

The CPU32 programming model consists of two groups of registers that correspond to the user and supervisor privilege levels. User programs can use only the registers of the user model. The supervisor programming model, which supplements the user programming model, is used by CPU32 system programmers who wish to protect sensitive operating system functions. The supervisor model is identical to that of the MC68010 and later processors.

The CPU32 has eight 32-bit data registers, seven 32-bit address registers, a 32-bit program counter, separate 32-bit supervisor and user stack pointers, a 16-bit status register, two alternate function code registers, and a 32-bit vector base register (see **Figure 5-2** and **Figure 5-3**).

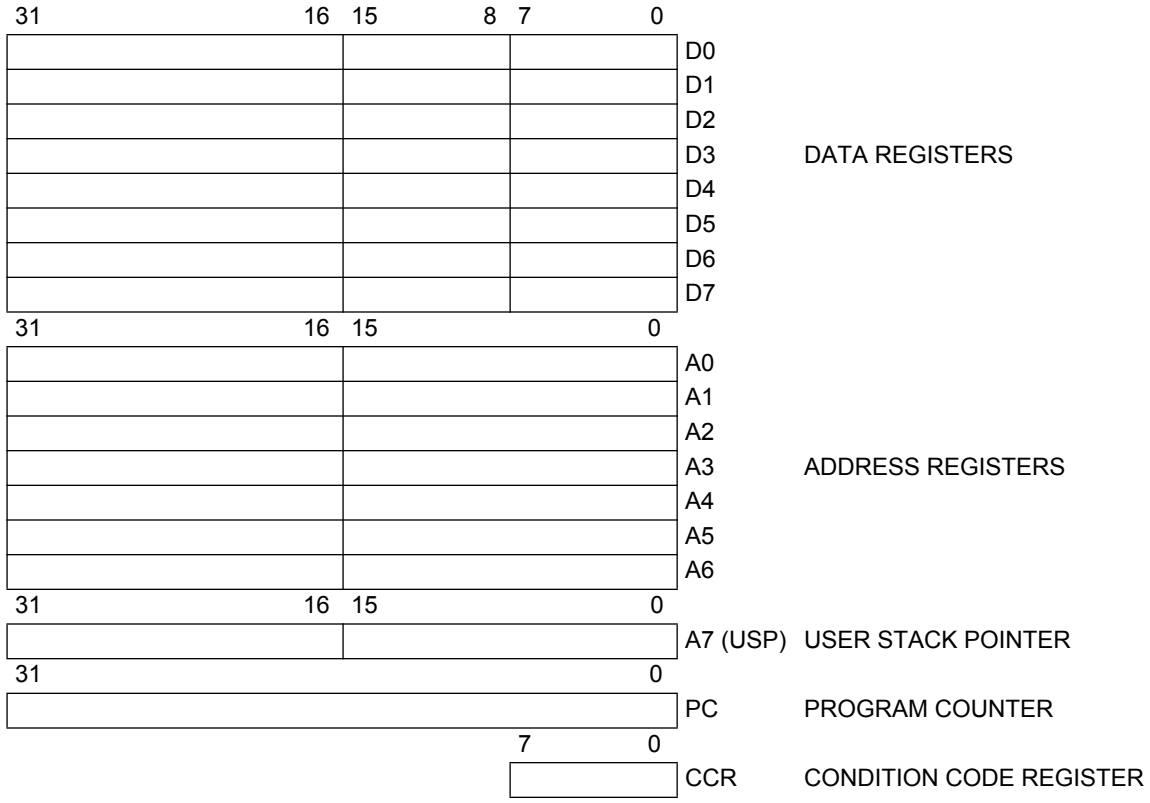


Figure 5-2 User Programming Model

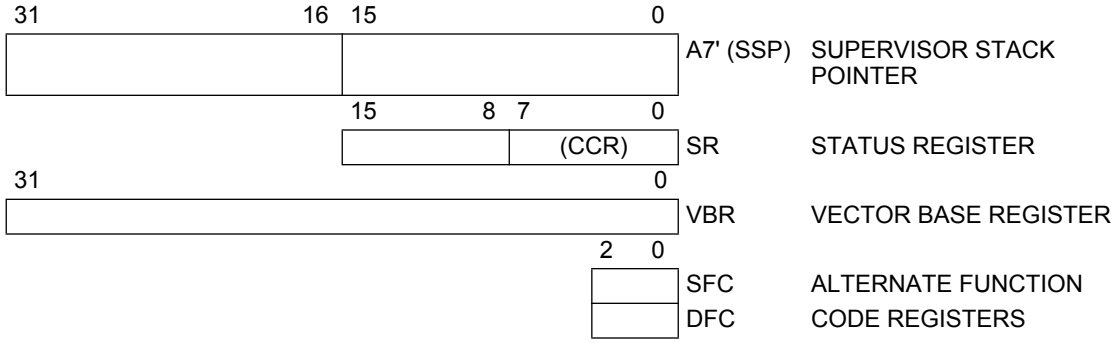


Figure 5-3 Supervisor Programming Model Supplement

5.2.1 Data Registers

The eight data registers can store data operands of 1, 8, 16, 32, and 64 bits and addresses of 16 or 32 bits. The following 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)

Each of data registers D7–D0 is 32 bits wide. Byte operands occupy the low-order 8 bits; word operands, the low-order 16 bits; and long-word operands, the entire 32 bits. When a data register is used as either a source or destination operand, only the appropriate low-order byte or word (in byte or word operations, respectively) is used or changed; the remaining high-order portion is unaffected. The least significant bit (LSB) of a long-word integer is addressed as bit zero, and the most significant bit (MSB) is addressed as bit 31. **Figure 5-4** shows the organization of various types of data in the data registers.

Quad-word data consists of two long words and represents the product of 32-bit multiply or the dividend of 32-bit divide operations (signed and unsigned). Quad-words may be organized in any two data registers without restrictions on order or pairing. There are no explicit instructions for the management of this data type, although the MOVEM instruction can be used to move a quad-word into or out of the registers.

Binary-coded decimal (BCD) data represents decimal numbers in binary form. CPU32 BCD instructions use a format in which a byte contains two digits. The four LSB contain the least significant digit, and the four MSB contain the most significant digit. The ABCD, SBCD, and NBCD instructions operate on two BCD digits packed into a single byte.

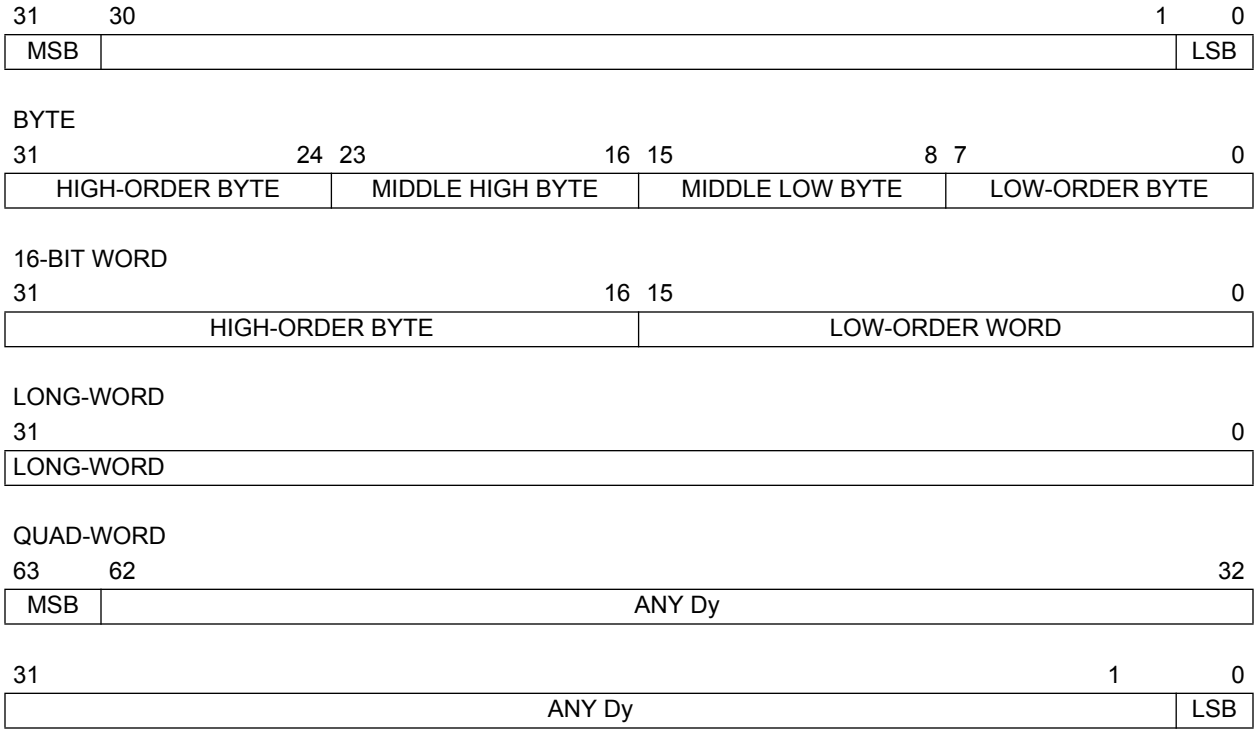


Figure 5-4 Data Organization in Data Registers

5.2.2 Address Registers

Each address register and stack pointer is 32 bits wide and holds a 32-bit address. Address registers cannot be used for byte-sized operands. Therefore, when an address register is used as a source operand, either the low-order word or the entire long-word operand is used, depending upon the operation size. When an address register is used as the destination operand, the entire register is affected, regardless of the operation size. If the source operand is a word size, it is sign-extended to 32 bits. Address registers are used primarily for addresses and to support address computation. The instruction set includes instructions that add to, subtract from, compare, and move the contents of address registers. **Figure 5-5** shows the organization of addresses in address registers.

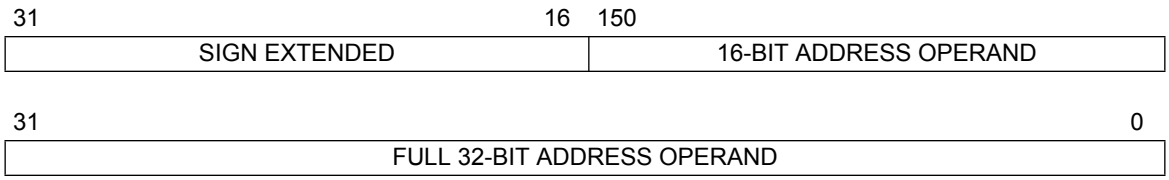


Figure 5-5 Address Organization in Address Registers

5.2.3 Program Counter

The PC contains the address of the next instruction to be executed by the CPU32. During instruction execution and exception processing, the processor automatically increments the contents of the PC or places a new value in the PC as appropriate.

5.2.4 Control Registers

The control registers described in this section contain control information for supervisor functions and vary in size. With the exception of the condition code register (the user portion of the status register), they are accessed only by instructions at the supervisor privilege level.

5.2.4.1 Status Register

The status register (SR) stores the processor status. It contains the condition codes that reflect the results of a previous operation and can be used for conditional instruction execution in a program. The condition codes are extend (X), negative (N), zero (Z), overflow (V), and carry (C). The user (low-order) byte containing the condition codes is the only portion of the SR information 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. The upper byte of this register includes the interrupt priority (IP) mask (three bits), two bits for placing the processor in one of two tracing modes or disabling tracing, and the supervisor/user bit for placing the processor at the desired privilege level.

Undefined bits in the status register are reserved by Freescale for future definition. The undefined bits are read as zeros and should be written as zeros for future compatibility.

All operations to the SR and CCR are word-size operations, but for all CCR operations, the upper byte is read as all zeros and is ignored when written, regardless of privilege level.

Refer to **APPENDIX D REGISTER SUMMARY** for bit/field definitions and a diagram of the status register.

5.2.4.2 Alternate Function Code Registers

Alternate function code registers (SFC and DFC) contain 3-bit function codes. Function codes can be considered extensions of the 24-bit linear address that optionally provide as many as eight 16-Mbyte address spaces. The processor automatically generates function codes to select address spaces for data and programs at the user and supervisor privilege levels and to select a CPU address space used for processor functions (such as breakpoint and interrupt acknowledge cycles).

Registers SFC and DFC are used by the MOVES instruction to specify explicitly the function codes of the memory address. The MOVEC instruction is used to transfer values to and from the alternate function code registers. This is a long-word transfer; the upper 29 bits are read as zeros and are ignored when written.

5.2.5 Vector Base Register (VBR)

The VBR contains the base address of the 1024-byte exception vector table, consisting of 256 exception vectors. Exception vectors contain the memory addresses of routines that begin execution at the completion of exception processing. Refer to **5.9 Exception Processing** for more information on the VBR and exception processing.

5.3 Memory Organization

Memory is organized on a byte-addressable basis in which lower addresses correspond to higher order bytes. For example, the address N of a long-word data item corresponds to the address of the most significant byte of the highest order word. The address of the most significant byte of the low-order word is N + 2, and the address of the least significant byte of the long word is N + 3. The CPU32 requires long-word and word data and instructions to be aligned on word boundaries (refer to **Figure 5-6**). Data misalignment is not supported.

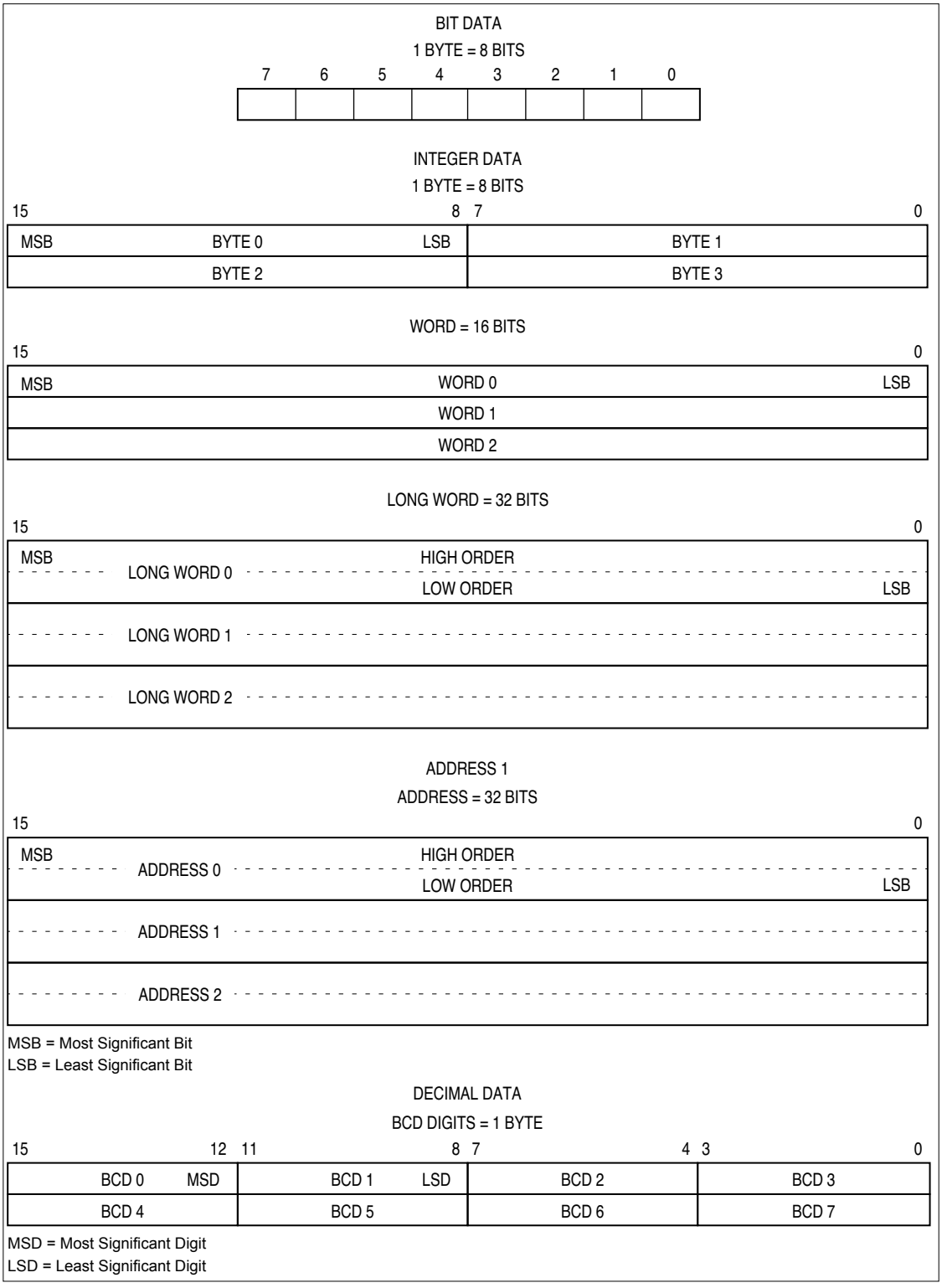


Figure 5-6 Memory Operand Addressing

5.4 Virtual Memory

The full addressing range of the CPU32 on the MC68331 is 16 Mbytes in each of eight address spaces. Even though most systems implement a smaller physical memory, the system can be made to appear to have a full 16 Mbytes of memory available to each user program by using virtual memory techniques.

A system that supports virtual memory has a limited amount of high-speed physical memory that can be accessed directly by the processor and maintains an image of a much larger virtual memory on a secondary storage device. When the processor attempts to access a location in the virtual memory map that is not resident in physical memory, a page fault occurs. The access to that location is temporarily suspended while the necessary data is fetched from secondary storage and placed in physical memory. The suspended access is then restarted or continued.

The CPU32 uses instruction restart, which requires that only a small portion of the internal machine state be saved. After correcting the fault, the machine state is restored, and the instruction is fetched and started again. This process is completely transparent to the application program.

5.5 Addressing Modes

Addressing in the CPU32 is register-oriented. Most instructions allow the results of the specified operation to be placed either in a register or directly in memory. There is no need for extra instructions to store register contents in memory.

There are seven basic addressing modes:

- Register Direct
- Register Indirect
- Register Indirect with Index
- Program Counter Indirect with Displacement
- Program Counter Indirect with Index
- Absolute
- Immediate

The register indirect addressing modes include postincrement, predecrement, and offset capability. The program counter indirect 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, and/or program counter.

5.6 Processing States

The processor is always in one of four processing states: normal, exception, halted, or background. The normal processing state is associated with instruction execution; the bus is used to fetch instructions and operands and to store results.

The exception processing state is associated with interrupts, trap instructions, tracing, and other exception conditions. The exception may be internally generated explicitly by an instruction or by an unusual condition arising during the execution of an instruction. Exception processing can be forced externally by an interrupt, a bus error, or a

The halted processing state is an indication of catastrophic hardware failure. For example, if during the exception processing of a bus error another bus error occurs, the processor assumes that the system is unusable and halts.

The background processing state is initiated by breakpoints, execution of special instructions, or a double bus fault. Background processing is enabled by pulling $\overline{\text{BKPT}}$ low during $\overline{\text{RESET}}$. Background processing allows interactive debugging of the system via a simple serial interface.

5.7 Privilege Levels

The processor operates at one of two levels of privilege: user or supervisor. Not all instructions are permitted to execute at the user level, but all instructions are available at the supervisor level. Effective use of privilege level can protect system resources from uncontrolled access. The state of the S bit in the status register determines the privilege level and whether the user stack pointer (USP) or supervisor stack pointer (SSP) is used for stack operations.

5.8 Instructions

The CPU32 instruction set is summarized in **Table 5-1**. The instruction set of the CPU32 is very similar to that of the MC68020. Two new instructions have been added to facilitate controller applications: low-power stop (LPSTOP) and table lookup and interpolate (TBLS, TBLSN, TBLU, TBLUN).

The following MC68020 instructions are not implemented on the CPU32:

BFxxx	—	Bit Field Instructions (BFCHG, BFCLR, BFEXTS, BFEXTU, BFFFO, BFINS, BFSET, BFTST)
CALLM, RTM	—	Call Module, Return Module
CAS, CAS2	—	Compare and Swap (Read-Modify-Write Instructions)
cpxxx	—	Coprocessor Instructions (cpBcc, cpDBcc, cpGEN, cpRESTORE, cpSAVE, cpScc, cpTRAPcc)
PACK, UNPK	—	Pack, Unpack BCD Instructions
Memory	—	Memory Indirect Addressing Modes

The CPU32 traps on unimplemented instructions or illegal effective addressing modes, allowing user-supplied code to emulate unimplemented capabilities or to define special purpose functions. However, Freescale reserves the right to use all currently unimplemented instruction operation codes for future M68000 core enhancements.

Table 5-1 Instruction Set Summary

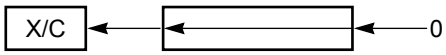
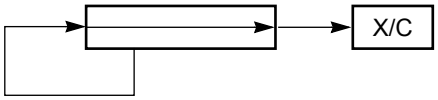
Instruction	Syntax	Operand Size	Operation
ABCD	Dn, Dn – (An), – (An)	8 8	$Source_{10}$ + $Destination_{10}$ + X ⇒ Destination
ADD	Dn, <ea> <ea>, Dn	8, 16, 32 8, 16, 32	Source + Destination ⇒ Destination
ADDA	<ea>, An	16, 32	Source + Destination ⇒ Destination
ADDI	#<data>, <ea>	8, 16, 32	Immediate data + 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 SR[1]	#<data>, SR	16	Source · SR ⇒ SR
ASL	Dn, Dn #<data>, Dn Í	8, 16, 32 8, 16, 32 16	
ASR	Dn, Dn #<data>, Dn Í	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	$\overline{(\text{<bit number> of destination})}$ ⇒ Z ⇒ bit of destination
BCLR	Dn, <ea> #<data>, <ea>	8, 32 8, 32	$\overline{(\text{<bit number> of destination})}$ ⇒ Z; 0 ⇒ bit of destination
BGND	none	none	If background mode enabled, then enter background mode, else format/vector offset ⇒ – (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	$\overline{(\text{<bit number> of destination})}$ ⇒ 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	$\overline{(\text{<bit number> of destination})}$ ⇒ 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	Í	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
CMPA	<ea>, An	16, 32	(Destination – Source), CCR shows results
CMPI	#<data>, <ea>	8, 16, 32	(Destination – Data), CCR shows results
CMPM	(An) +, (An) +	8, 16, 32	(Destination – Source), CCR shows results
	<ea>, Rn	8, 16, 32	Lower bound ≤ Rn ≤ Upper bound, CCR shows result

Table 5-1 Instruction Set Summary

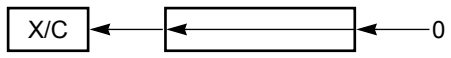
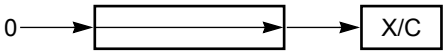
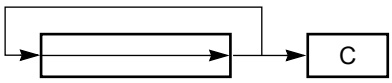
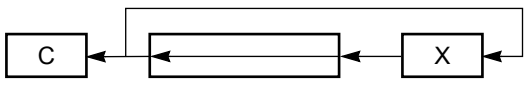
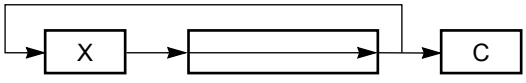
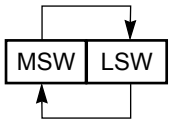
DBcc	Dn, <label>	16	If condition false, then Dn – 1 ⇒ PC; if Dn ∂ (– 1), then PC + d ⇒ PC
DIVS/DIVU	<ea>, Dn	32/16 ⇒ 16: 16	Destination / Source ⇒ Destination (signed or unsigned)
DIVSL/DIVUL	<ea>, Dr : Dq <ea>, Dq <ea>, Dr : Dq	64/32 ⇒ 32 : 32 32/32 ⇒ 32 32/32 ⇒ 32 : 32	Destination / Source ⇒ Destination (signed or unsigned)
EOR	Dn, <ea>	8, 16, 32	Source x Destination ⇒ Destination
EORI	#<data>, <ea>	8, 16, 32	Data x Destination ⇒ Destination
EORI to CCR	#<data>, CCR	8	Source x CCR ⇒ CCR
EORI to SR[1]	#<data>, SR	16	Source x 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	Í	none	Destination ⇒ PC
JSR	Í	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[1]	#<data>	none	Data ⇒ SR; interrupt mask ⇒ EBI; STOP
LSL	Dn, Dn #<data>, Dn Í	8, 16, 32 8, 16, 32 16	
LSR	Dn, Dn #<data>, Dn Í	8, 16, 32 8, 16, 32 16	
MOVE	<ea>, <ea>	8, 16, 32	Source ⇒ Destination
MOVEA	<ea>, An	16, 32 ⇒ 32	Source ⇒ Destination
MOVEA[1]	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[1]	SR, <ea>	16	SR ⇒ Destination
ROR	Dn, Dn #<data>, Dn Í	8, 16, 32 8, 16, 32 16	
ROXL	Dn, Dn #<data>, Dn Í	8, 16, 32 8, 16, 32 16	
ROXR	Dn, Dn #<data>, Dn Í	8, 16, 32 8, 16, 32 16	
RTD	#<d>	16	(SP) ⇒ PC; SP + 4 + d ⇒ SP
RTE[1]	none	none	(SP) ⇒ SR; SP + 2 ⇒ SP; (SP) ⇒ PC; SP + 4 ⇒ SP; restore stack according to format

Table 5-1 Instruction Set Summary

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	$Destination_{10}$ – $Source_{10}$ – X ⇒ Destination
Scc	Í	8	If condition true, then destination bits are set to 1; else, destination bits are cleared to 0
STOP[1]	#<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	
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	Í	8, 16, 32	Source – 0, to set condition codes
UNLK	An	32	An ⇒ SP; (SP) ⇒ An, SP + 4 ⇒ SP

NOTE:
1. Privileged instruction.

5.8.1 M68000 Family Compatibility

It is the philosophy of the M68000 family that all user-mode programs can execute unchanged on a more advanced processor, and supervisor-mode programs and exception handlers should require only minimal alteration.

The CPU32 can be thought of as an intermediate member of the M68000 Family. Object code from an MC68000 or MC68010 may be executed on the CPU32, and many of the instruction and addressing mode extensions of the MC68020 are also supported. Refer to the CPU32 reference manual for a detailed comparison of the CPU32 and MC68020 instruction set.

5.8.2 Special Control Instructions

Low power stop (LPSTOP) and table lookup and interpolate (TBL) instructions have been added to the MC68000 instruction set for use in controller applications.

5.8.2.1 Low Power Stop (LPSTOP)

In applications where power consumption is a consideration, the CPU32 forces the device into a low power standby mode when immediate processing is not required. The low power stop mode is entered by executing the LPSTOP instruction. The processor remains in this mode until a user-specified (or higher) interrupt level or reset occurs.

5.8.2.2 Table Lookup and Interpolate (TBL)

To maximize throughput for real-time applications, reference data is often precalculated and stored in memory for quick access. Storage of many data points can require an inordinate amount of memory. The table instruction requires that only a sample of data points be stored, reducing memory requirements. The TBL instruction recovers intermediate values using linear interpolation. Results can be rounded with a round-to-nearest algorithm.

5.9 Exception Processing

An exception is a special condition that preempts normal processing. Exception processing is the transition from normal mode program execution to execution of a routine that deals with an exception.

5.9.1 Exception Vectors

An exception vector is the address of a routine that handles an exception. The vector base register (VBR) contains the base address of a 1024-byte exception vector table, which consists of 256 exception vectors. Sixty-four vectors are defined by the processor, and 192 vectors are reserved for user definition as interrupt vectors. Except for the reset vector, each vector in the table is one long word in length. The reset vector is two long words in length. Refer to **Table 5-2** for information on vector assignment.

CAUTION

Because there is no protection on the 64 processor-defined vectors, external devices can access vectors reserved for internal purposes. This practice is strongly discouraged.

All exception vectors, except the reset vector, are located in supervisor data space. The reset vector is located in supervisor program space. Only the initial reset vector is fixed in the processor memory map. When initialization is complete, there are no fixed assignments. Since the VBR stores the vector table base address, the table can be located anywhere in memory. It can also be dynamically relocated for each task executed by an operating system.

Table 5-2 Exception Vector Assignments

Vector Number	Vector Offset			Assignment
	Dec	Hex	Space	
0	0	000	SP	Reset: Initial Stack Pointer
1	4	004	SP	Reset: Initial Program Counter
2	8	008	SD	Bus Error
3	12	00C	SD	Address Error
4	16	010	SD	Illegal Instruction
5	20	014	SD	Zero Division
6	24	018	SD	CHK, CHK2 Instructions
7	28	01C	SD	TRAPcc, TRAPV Instructions
8	32	020	SD	Privilege Violation
9	36	024	SD	Trace
10	40	028	SD	Line 1010 Emulator
11	44	02C	SD	Line 1111 Emulator
12	48	030	SD	Hardware Breakpoint
13	52	034	SD	(Reserved, Coprocessor Protocol Violation)
14	56	038	SD	Format Error and Uninitialized Interrupt
15	60	03C	SD	Format Error and Uninitialized Interrupt
16–23	64 92	040 05C	SD	(Unassigned, Reserved)
24	96	060	SD	Spurious Interrupt
25	100	064	SD	Level 1 Interrupt Autovector
26	104	068	SD	Level 2 Interrupt Autovector
27	108	06C	SD	Level 3 Interrupt Autovector
28	112	070	SD	Level 4 Interrupt Autovector
29	116	074	SD	Level 5 Interrupt Autovector
30	120	078	SD	Level 6 Interrupt Autovector
31	124	07C	SD	Level 7 Interrupt Autovector
32–47	128 188	080 0BC	SD	Trap Instruction Vectors (0–15)
48–58	192 232	0C0 0E8	SD	(Reserved, Coprocessor)
59–63	236 252	0EC 0FC	SD	(Unassigned, Reserved)
64–255	256 1020	100 3FC	SD	User Defined Vectors (192)

Each vector is assigned an 8-bit number. Vector numbers for some exceptions are obtained from an external device; others are supplied by the processor. The processor multiplies the vector number by four to calculate vector offset, then adds the offset to the contents of the VBR. The sum is the memory address of the vector.

5.9.2 Types of Exceptions

An exception can be caused by internal or external events.

An internal exception can be generated by an instruction or by an error. The TRAP, TRAPcc, TRAPV, BKPT, CHK, CHK2, RTE, and DIV instructions can cause exceptions during normal execution. Illegal instructions, instruction fetches from odd addresses, word or long-word operand accesses from odd addresses, and privilege violations also cause internal exceptions.

Sources of external exception include interrupts, breakpoints, bus errors, and reset requests. Interrupts are peripheral device requests for processor action. Breakpoints are used to support development equipment. Bus error and reset are used for access control and processor restart.

5.9.3 Exception Processing Sequence

For all exceptions other than a reset exception, exception processing occurs in the following sequence. Refer to **4.6 Reset** for details of reset processing.

As exception processing begins, the processor makes an internal copy of the status register. After the copy is made, the processor state bits in the status register are changed — the S bit is set, establishing supervisor access level, and bits T1 and T0 are cleared, disabling tracing. For reset and interrupt exceptions, the interrupt priority mask is also updated.

Next, the exception number is obtained. For interrupts, the number is fetched from CPU space $F (the bus cycle is an interrupt acknowledge). For all other exceptions, internal logic provides a vector number.

Next, current processor status is saved. An exception stack frame is created and placed on the supervisor stack. All stack frames contain copies of the status register and the program counter for use by RTE. The type of exception and the context in which the exception occurs determine what other information is stored in the stack frame.

Finally, the processor prepares to resume normal execution of instructions. The exception vector offset is determined by multiplying the vector number by four, and the offset is added to the contents of the VBR to determine displacement into the exception vector table. The exception vector is loaded into the program counter. If no other exception is pending, the processor will resume normal execution at the new address in the PC.

5.10 Development Support

The following features have been implemented on the CPU32 to enhance the instrumentation and development environment:

- M68000 Family Development Support
- Background Debugging Mode
- Deterministic Opcode Tracking
- Hardware Breakpoints

5.10.1 M68000 Family Development Support

All M68000 Family members include features to facilitate applications development. These features include the following:

Trace on Instruction Execution — M68000 Family processors include an instruction-by-instruction tracing facility as an aid to program development. The MC68020, MC68030, MC68040, and CPU32 also allow tracing only of those instructions

causing a change in program flow. In the trace mode, a trace exception is generated after an instruction is executed, allowing a debugger program to monitor the execution of a program under test.

Breakpoint Instruction — An emulator may insert software breakpoints into the target code to indicate when a breakpoint has occurred. On the MC68010, MC68020, MC68030, and CPU32, this function is provided via illegal instructions, $4848–$484F, to serve as breakpoint instructions.

Unimplemented Instruction Emulation — During instruction execution, when an attempt is made to execute an illegal instruction, an illegal instruction exception occurs. Unimplemented instructions (F-line, A-line,...) utilize separate exception vectors to permit efficient emulation of unimplemented instructions in software.

5.10.2 Background Debugging Mode

Microcomputer systems generally provide a debugger, implemented in software, for system analysis at the lowest level. The background debugging mode (BDM) on the CPU32 is unique in that the debugger has been implemented in CPU microcode.

BDM incorporates a full set of debugging options: registers can be viewed or altered, memory can be read or written to, and test features can be invoked.

A resident debugger simplifies implementation of an in-circuit emulator. In a common setup (see **Figure 5-7**), emulator hardware replaces the target system processor. A complex, expensive pod-and-cable interface provides a communication path between the target system and the emulator.

By contrast, an integrated debugger supports use of a bus state analyzer (BSA) for in-circuit emulation. The processor remains in the target system (see **Figure 5-8**) and the interface is simplified. The BSA monitors target processor operation and the on-chip debugger controls the operating environment. Emulation is much “closer” to target hardware, and many interfacing problems (e.g., limitations on high-frequency operation, AC and DC parametric mismatches, and restrictions on cable length) are minimized.

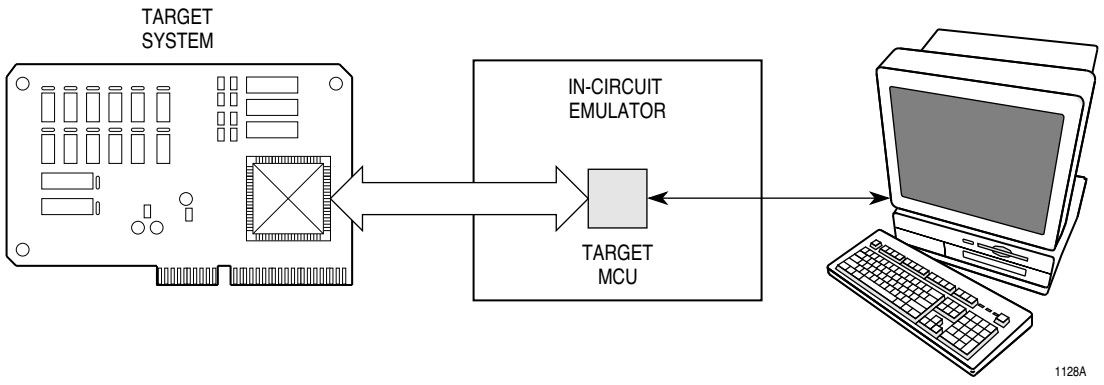


Figure 5-7 Common in-Circuit Emulator Diagram

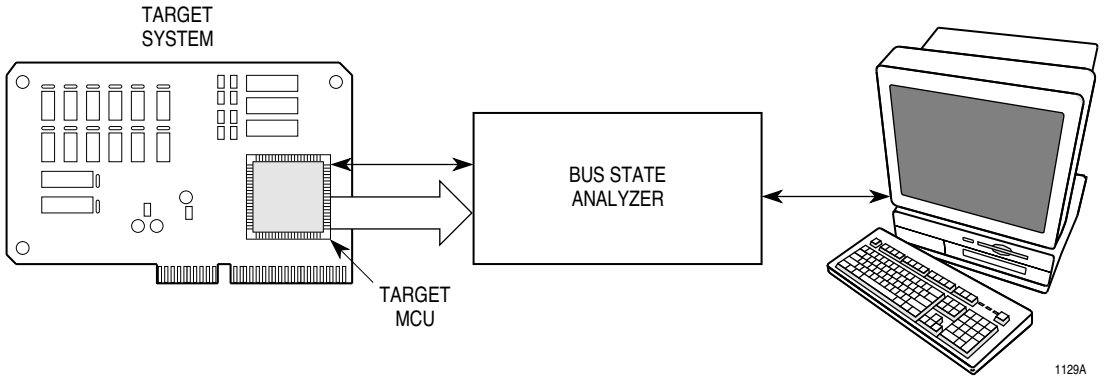


Figure 5-8 Bus State Analyzer Configuration

5.10.2.1 Enabling BDM

Accidentally entering BDM in a non-development environment can lock up the CPU32 when the serial command interface is not available. For this reason, BDM is enabled during reset via the breakpoint ($\overline{\text{BKPT}}$) signal.

BDM operation is enabled when $\overline{\text{BKPT}}$ is asserted (low), at the rising edge of $\overline{\text{RESET}}$. BDM remains enabled until the next system reset. A high $\overline{\text{BKPT}}$ signal on the trailing edge of $\overline{\text{RESET}}$ disables BDM. $\overline{\text{BKPT}}$ is latched again on each rising transition of $\overline{\text{RESET}}$. $\overline{\text{BKPT}}$ is synchronized internally, and must be held low for at least two clock cycles prior to negation of $\overline{\text{RESET}}$.

BDM enable logic must be designed with special care. If hold time on $\overline{\text{BKPT}}$ extends into the first bus cycle following reset, the bus cycle could inadvertently be tagged with a breakpoint. Refer to the *SIM Reference Manual* (SIMRM/AD) for timing information.

5.10.2.2 BDM Sources

When BDM is enabled, any of several sources can cause the transition from normal mode to BDM. These sources include external breakpoint hardware, the BGND instruction, a double bus fault, and internal peripheral breakpoints. If BDM is not enabled when an exception condition occurs, the exception is processed normally. **Table 5-3** summarizes the processing of each source for both enabled and disabled cases. As shown in **Table 5-3**, the BKPT instruction never causes a transition into BDM.

Table 5-3 BDM Source Summary

Source	BDM Enabled	BDM Disabled
$\overline{\text{BKPT}}$	Background	Breakpoint Exception
Double Bus Fault	Background	Halted
BGND Instruction	Background	Illegal Instruction
BKPT Instruction	Opcode Substitution/ Illegal Instruction	Opcode Substitution/ Illegal Instruction

5.10.2.2.1 External $\overline{\text{BKPT}}$ Signal

Once enabled, BDM is initiated whenever assertion of $\overline{\text{BKPT}}$ is acknowledged. If BDM is disabled, a breakpoint exception (vector $0C) is acknowledged. The $\overline{\text{BKPT}}$ input has the same timing relationship to the data strobe trailing edge as does read cycle data. There is no breakpoint acknowledge bus cycle when BDM is entered.

5.10.2.2.2 BGND Instruction

An illegal instruction, $4AFA, is reserved for use by development tools. The CPU32 defines $4AFA (BGND) to be a BDM entry point when BDM is enabled. If BDM is disabled, an illegal instruction trap is acknowledged.

5.10.2.2.3 Double Bus Fault

The CPU32 normally treats a double bus fault, or two bus faults in succession, as a catastrophic system error, and halts. When this condition occurs during initial system debug (a fault in the reset logic), further debugging is impossible until the problem is corrected. In BDM, the fault can be temporarily bypassed, so that the origin of the fault can be isolated and eliminated.

5.10.2.2.4 Peripheral Breakpoints

CPU32 peripheral breakpoints are implemented in the same way as external breakpoints — peripherals request breakpoints by asserting the $\overline{\text{BKPT}}$ signal. Consult the appropriate peripheral user's manual for additional details on the generation of peripheral breakpoints.

5.10.2.3 Entering BDM

When the processor detects a breakpoint or a double bus fault, or decodes a BGND instruction, it suspends instruction execution and asserts the FREEZE output. This is the first indication that the processor has entered BDM. Once FREEZE has been asserted, the CPU enables the serial communication hardware and awaits a command.

The CPU writes a unique value indicating the source of BDM transition into temporary register A (ATEMP) as part of the process of entering BDM. A user can poll ATEMP and determine the source (see **Table 5-4**) by issuing a read system register command (RSREG). ATEMP is used in most debugger commands for temporary storage — it is imperative that the RSREG command be the first command issued after transition into BDM.

Table 5-4 Polling the BDM Entry Source

Source	ATEMP[31:16]	ATEMP[15:0]
Double Bus Fault	SSW*	$FFFF
BGND Instruction	$0000	$0001
Hardware Breakpoint	$0000	$0000

*Special status word (SSW) is described in detail in the *CPU32 Reference Manual* (CPU32RM/AD).

A double bus fault during initial stack pointer/program counter (SP/PC) fetch sequence is distinguished by a value of $FFFFFFFF in the current instruction PC. At no other time will the processor write an odd value into this register.

5.10.2.4 BDM Commands

Commands consist of one 16-bit operation word and can include one or more 16-bit extension words. Each incoming word is read as it is assembled by the serial interface. The microcode routine corresponding to a command is executed as soon as the command is complete. Result operands are loaded into the output shift register to be shifted out as the next command is read. This process is repeated for each command until the CPU returns to normal operating mode. **Table 5-5** is a summary of background mode commands.

Table 5-5 Background Mode Command Summary

Command	Mnemonic	Description
Read D/A Register	RDREG/RAREG	Read the selected address or data register and return the results via 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. An initial WRITE is executed to set up the starting address of the block and supply the first operand. Subsequent operands are written with the FILL command.
Resume Execution	GO	The pipe is flushed and re-filled 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 $\overline{\text{RESET}}$ for 512 clock cycles. The CPU is not reset by this command. Synonymous with the CPU $\overline{\text{RESET}}$ instruction.
No Operation	NOP	NOP performs no operation and may be used as a null command.

5.10.2.5 Background Mode Registers

BDM processing uses three special purpose registers to keep track of program context during development. A description of each follows.

5.10.2.5.1 Fault Address Register (FAR)

The FAR contains the address of the faulting bus cycle immediately following a bus or address error. This address remains available until overwritten by a subsequent bus cycle. Following a double bus fault, the FAR contains the address of the last bus cycle. The address of the first fault (if there was one) is not visible to the user.

5.10.2.5.2 Return Program Counter (RPC)

The RPC points to the location where fetching will commence after transition from background mode to normal mode. This register should be accessed to change the flow of a program under development. Changing the RPC to an odd value will cause an address error when normal mode prefetching begins.

5.10.2.5.3 Current Instruction Program Counter (PCC)

The PCC holds a pointer to the first word of the last instruction executed prior to transition into background mode. Due to instruction pipelining, the instruction pointed to may not be the instruction which caused the transition. An example is a breakpoint on a released write. The bus cycle may overlap as many as two subsequent instructions before stalling the instruction sequencer. A breakpoint asserted during this cycle will not be acknowledged until the end of the instruction executing at completion of the bus cycle. PCC will contain $00000001 if BDM is entered via a double bus fault immediately out of reset.

5.10.2.6 Returning from BDM

BDM is terminated when a resume execution (GO) or call user code (CALL) command is received. Both GO and CALL flush the instruction pipeline and refetch instructions from the location pointed to by the RPC.

The return PC and the memory space referred to by the status register SUPV bit reflect any changes made during BDM. FREEZE is negated prior to initiating the first prefetch. Upon negation of FREEZE, the serial subsystem is disabled, and the signals revert to $\overline{\text{IPIPE}}$/$\overline{\text{IFETCH}}$ functionality.

5.10.2.7 Serial Interface

Communication with the CPU32 during BDM occurs via a dedicated serial interface, which shares pins with other development features. **Figure 5-9** is a block diagram of the interface. The $\overline{\text{BKPT}}$ signal becomes the serial clock (DSCLK); serial input data (DSI) is received on $\overline{\text{IFETCH}}$, and serial output data (DSO) is transmitted on $\overline{\text{IPIPE}}$.

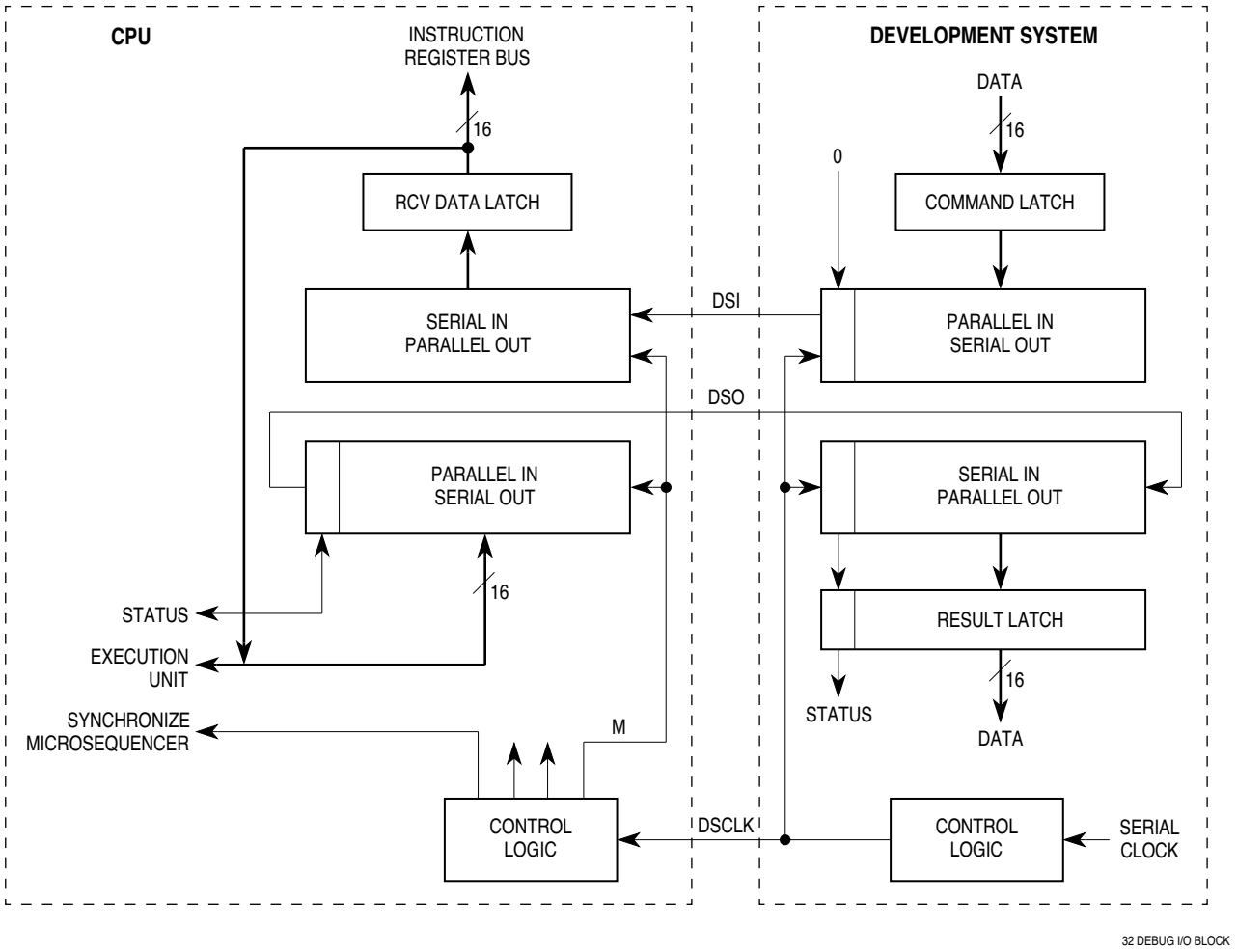


Figure 5-9 Debug Serial I/O Block Diagram

The serial interface uses a full-duplex synchronous protocol similar to the serial peripheral interface (SPI) protocol. The development system serves as the master of the serial link since it is responsible for the generation of DSCLK. If DSCLK is derived from the CPU32 system clock, development system serial logic is unhindered by the operating frequency of the target processor. Operable frequency range of the serial clock is from DC to one-half the processor system clock frequency.

The serial interface operates in full-duplex mode — data is transmitted and received simultaneously by both master and slave devices. In general, data transitions occur on the falling edge of DSCLK and are stable by the following rising edge of DSCLK. Data is transmitted MSB first, and is latched on the rising edge of DSCLK.

The serial data word is 17 bits wide — 16 data bits and a status/control bit. Bit 16 indicates the status of CPU-generated messages as shown in **Table 5-6**.

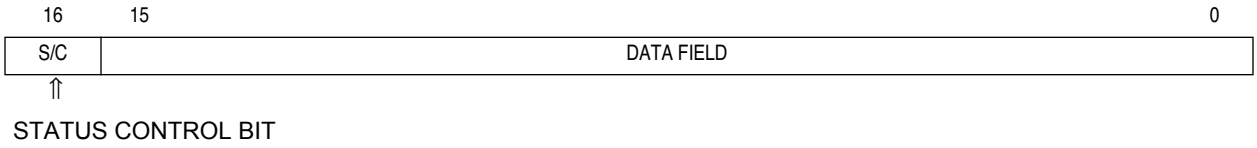


Figure 5-10 BDM Serial Data Word

Table 5-6 CPU Generated Message Encoding

Bit 16	Data	Message Type
0	xxxx	Valid Data Transfer
0	FFFF	Command Complete; Status OK
1	0000	Not Ready with Response; Come Again
1	0001	$\overline{\text{BERR}}$ Terminated Bus Cycle; Data Invalid
1	FFFF	Illegal Command

Command and data transfers initiated by the development system should clear bit 16. The current implementation ignores this bit; however, Freescale reserves the right to use this bit for future enhancements.

5.10.3 Recommended BDM Connection

In order to provide for use of development tools when an MCU is installed in a system, Freescale recommends that appropriate signal lines be routed to a male Berg connector or double-row header installed on the circuit board with the MCU, as shown in the following figure.

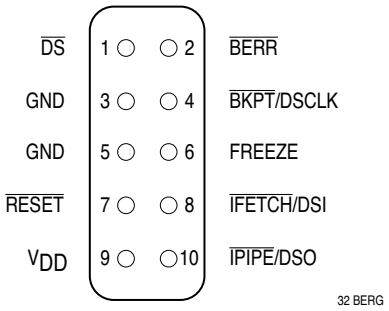


Figure 5-11 BDM Connector Pinout

5.10.4 Deterministic Opcode Tracking

CPU32 function code outputs are augmented by two supplementary signals to monitor the instruction pipeline. The instruction pipe ($\overline{\text{IPIPE}}$) output indicates the start of each new instruction and each mid-instruction pipeline advance. The instruction fetch ($\overline{\text{IFETCH}}$) output identifies the bus cycles in which the operand is loaded into the instruction pipeline. Pipeline flushes are also signaled with $\overline{\text{IFETCH}}$. Monitoring these two signals allows a bus analyzer to synchronize itself to the instruction stream and monitor its activity.

5.10.5 On-Chip Breakpoint Hardware

An external breakpoint input and on-chip breakpoint hardware allow a breakpoint trap on any memory access. Off-chip address comparators preclude breakpoints unless show cycles are enabled. Breakpoints on instruction prefetches that are ultimately flushed from the instruction pipeline are not acknowledged; operand breakpoints are always acknowledged. Acknowledged breakpoints initiate exception processing at the address in exception vector number 12, or alternately enter background mode.

5.11 Loop Mode Instruction Execution

The CPU32 has several features that provide efficient execution of program loops. One of these features is the DBcc looping primitive instruction. To increase the performance of the CPU32, a loop mode has been added to the processor. The loop mode is used by any single word instruction that does not change the program flow. Loop mode is implemented in conjunction with the DBcc instruction. **Figure 5-12** shows the required form of an instruction loop for the processor to enter loop mode.

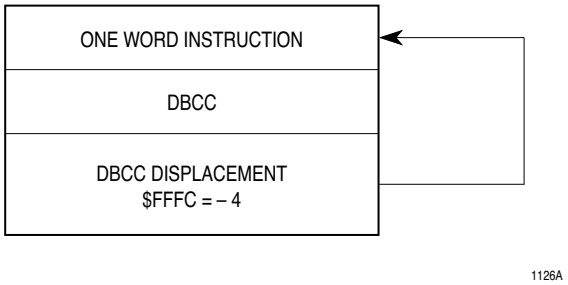


Figure 5-12 Loop Mode Instruction Sequence

The loop mode is entered when the DBcc instruction is executed, and the loop displacement is –4. Once in loop mode, the processor performs only the data cycles associated with the instruction and suppresses all instruction fetches. The termination condition and count are checked after each execution of the data operations of the looped instruction. The CPU32 automatically exits the loop mode on interrupts or other exceptions. All single word instructions that do not cause a change of flow can be looped.

SECTION 6 QUEUED SERIAL MODULE

This section is an overview of queued serial module (QSM) function. Refer to the *QSM Reference Manual* (QSMRM/AD) for complete information about the QSM.

6.1 General

The QSM contains two serial interfaces, the queued serial peripheral interface (QSPI) and the serial communication interface (SCI). **Figure 6-1** is a block diagram of the QSM.

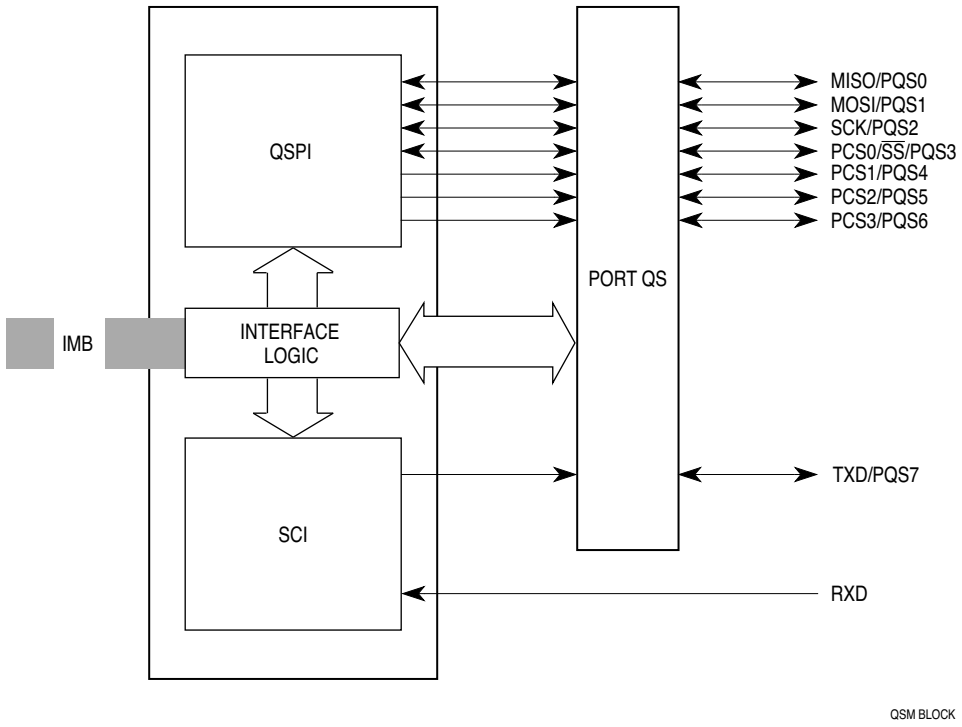


Figure 6-1 QSM Block Diagram

The QSPI provides easy peripheral expansion or interprocessor communication through a full-duplex, synchronous, three-line bus. Four programmable peripheral chip selects can select up to 16 peripheral devices. A self-contained RAM queue allows up to sixteen serial transfers of eight to sixteen 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, for efficient interfacing to A/D converters.

The SCI provides a standard nonreturn to zero (NRZ) mark/space format. It will operate 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 64 to 524 kbaud with a 16.78-MHz system clock, or 110 to 655 kbaud with a 20.97-MHz system clock. Word length of either eight or nine bits can be selected. Optional parity generation and detection provide either even or odd parity check capabil-

ity. 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.

6.2 QSM Registers and Address Map

There are four types of QSM registers: QSM global registers, QSM pin control registers, QSPI registers, and SCI registers. Global registers and pin control registers are discussed in **6.2.1 QSM Global Registers** and **6.2.2 QSM Pin Control Registers**. QSPI and SCI registers are discussed in **6.3 Queued Serial Peripheral Interface** and **6.4 Serial Communication Interface**. Writes to unimplemented register bits have no meaning or effect, and reads from unimplemented bits always return a logic zero value.

The QSM address map includes the QSM registers and the QSPI RAM. The module mapping (MM) bit in the SIM configuration register (SIMCR) defines the most significant bit (ADDR23) of the IMB address for each module in the MCU.

Refer to **APPENDIX D REGISTER SUMMARY** for a QSM address map and register bit/field definitions. **SECTION 4 SYSTEM INTEGRATION MODULE** contains more information about how the state of MM affects the system.

6.2.1 QSM Global Registers

The QSM configuration register (QSMCR) contains parameters for interfacing to the CPU32 and the intermodule bus. The QSM test register (QTEST) is used during factory test of the QSM. The QSM interrupt level register (QILR) determines the priority of interrupts requested by the QSM and the vector used when an interrupt is acknowledged. The QSM interrupt vector register (QIVR) contains the interrupt vector for both QSM submodules. QILR and QIVR are 8-bit registers located at the same word address. Refer to **APPENDIX D REGISTER SUMMARY** for register bit and field definitions.

6.2.1.1 Low-Power Stop Operation

When the STOP bit in the QSMCR is set, the system clock input to the QSM is disabled and the module enters a low-power operating state. QSMCR is the only register guaranteed to be readable while STOP is asserted. The QSPI RAM is not readable, but writes to RAM or any register are guaranteed valid while STOP is asserted. STOP can be set by the CPU and by reset.

System software must stop the QSPI and SCI before asserting STOP to prevent data corruption and simplify restart. Disable both SCI receiver and transmitter after transfers in progress are complete. Halt the QSPI by setting the HALT bit in SPCR3 and then setting STOP after the HALTA flag is set. Refer to **SECTION 4 SYSTEM INTEGRATION MODULE** for more information about low-power operation.

6.2.1.2 Freeze Operation

The freeze (FRZ[1:0]) bits in the QSMCR are used to determine what action is taken by the QSM when the IMB FREEZE signal is asserted. FREEZE is asserted when the CPU enters background debugging mode. At the present time, FRZ0 has no effect; setting FRZ1 causes the QSPI to halt on the first transfer boundary following FREEZE assertion. Refer to **SECTION 5 CENTRAL PROCESSING UNIT** for more information about background debugging mode.

6.2.1.3 QSM Interrupts

Both the QSPI and SCI can make interrupt requests on the IMB. Each has a separate interrupt request priority register, but a single vector register is used to generate exception vector numbers.

The values of the ILQSPI and ILSCI fields in the QILR determine the priority of QSPI and SCI interrupt requests. The values in these fields correspond to internal interrupt request signals $\overline{IRQ}$[7:1]. A value of %111 causes $\overline{IRQ7}$ to be asserted when a QSM interrupt request is made; lower field values cause corresponding lower-numbered interrupt request signals to be asserted. Setting field value to %000 disables interrupts. If ILQSPI and ILSCI have the same nonzero value, and the QSPI and SCI make simultaneous interrupt requests, the QSPI has priority.

When the CPU32 acknowledges an interrupt request, it places the value in the interrupt priority (IP) mask in the CPU status register on the address bus. The QSM compares IP mask value to request priority to determine whether it should contend for arbitration priority. Arbitration priority is determined by the value of the IARB field in the QSMCR. Each module that generates interrupts must have a nonzero IARB value. Arbitration is performed by means of serial assertion of IARB field bit values.

When the QSM wins interrupt arbitration, it responds to the CPU interrupt acknowledge cycle by placing an interrupt vector number on the data bus. The vector number is used to calculate displacement into the CPU32 exception vector table. SCI and QSPI vector numbers are generated from the value in the QIVR INTV field. The values of bits INTV[7:1] are the same for QSPI and SCI, but the value of INTV0 is supplied by the QSM when an interrupt request is made. INTV0 = 0 for SCI interrupt requests; INTV0 = 1 for QSPI requests.

At reset, INTV is initialized to $0F, the uninitialized interrupt vector number. To enable interrupt-driven serial communication, a user-defined vector number ($40–$FF) must be written to QIVR, and interrupt handler routines must be located at the addresses pointed to by the corresponding vector. CPU writes to INTV0 have no meaning or effect. Reads of INTV0 return a value of one.

Refer to **SECTION 5 CENTRAL PROCESSING UNIT** and **SECTION 4 SYSTEM INTEGRATION MODULE** for more information about exceptions and interrupts.

6.2.2 QSM Pin Control Registers

The QSM uses nine pins. Eight of the pins can be used for serial communication or for parallel I/O. Clearing a bit in the port QS pin assignment register (PQSPAR) assigns

the corresponding pin to general-purpose I/O; setting a bit assigns the pin to the QSPI. PQSPAR does not affect operation of the SCI.

The port QS data direction register (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. DDQS1 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. PQSPAR and DDRQS are 8-bit registers located at the same word address. **Table 6-1** is a summary of QSM pin functions.

The port QS data register (PORTQS) latches I/O data. Writes to PORTQS drive pins defined as outputs. PORTQS reads return data present on the pins when the read is made. To avoid driving undefined data, first write PORTQS, then configure DDRQS.

Table 6-1 QSM Pin Function

QSM Pin	Mode	DDRQS Bit	Bit State	Pin Function
MISO	Master	DDQS0	0	Serial Data Input to QSPI
			1	Disables Data Input
	Slave		0	Disables Data Output
			1	Serial Data Output from QSPI
MOSI	Master	DDQS1	0	Disables Data Output
			1	Serial Data Output from QSPI
	Slave		0	Serial Data Input to QSPI
			1	Disables Data Input
SCK[1]	Master	DDQS2	0	Disables Clock Output
			1	Clock Output from QSPI
	Slave		0	Clock Input to QSPI
			1	Disables Clock Input
PCS0/$\overline{SS}$	Master	DDQS3	0	Assertion Causes Mode Fault
			1	Chip-Select Output
	Slave		0	QSPI Slave Select Input
			1	Disables Select Input
PCS[3:1]	Master	DDQS[4:6]	0	Disables Chip-Select Output
			1	Chip-Select Output
	Slave		0	Inactive
			1	Inactive
TXD[2]	Transmit	DDQS7	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 the SPI serial clock SCK.
2. PQS7 is a digital I/O pin unless the SCI transmitter is enabled (TE in SCCR1 set), in which case it becomes SCI serial output TXD and DDRQS has no effect.

6.3 Queued Serial Peripheral Interface

The queued serial peripheral interface (QSPI) communicates with external devices through a synchronous serial bus. The QSPI is fully compatible with SPI systems found on other Freescale products, but has enhanced capabilities. The QSPI can per-

form full duplex three-wire or half duplex two-wire transfers. A variety of transfer rate, clocking, and interrupt-driven communication options are available.

Serial transfer of any number of bits from eight to sixteen can be specified. Programmable transfer length simplifies interfacing to a number of devices that require different data lengths.

An inter-transfer delay of 17 to 8192 system clocks can be specified (default is 17 system clocks). Programmable delay simplifies the interface to a number of devices that require different delays between transfers.

A dedicated 80-byte RAM is used to store received data, data to be transmitted, and a queue of commands. The CPU can access these locations directly. Serial peripherals can be treated like memory-mapped parallel devices.

The command queue allows the QSPI to perform up to 16 serial transfers without CPU intervention. Each queue entry contains all the information needed by the QSPI to independently complete one serial transfer.

A pointer identifies the queue location containing the command for the next serial transfer. Normally, the pointer address is incremented after each serial transfer, but the CPU can change the pointer value at any time. Multiple-task support can be provided by segmenting the queue.

The QSPI has four peripheral chip-select pins. Chip-select signals simplify interfacing by reducing CPU intervention. If chip-select signals are externally decoded, 16 independent select signals can be generated. Each chip-select pin can drive up to four independent peripherals, depending on loading.

Wraparound operating mode allows continuous execution of queued commands. In wraparound mode, newly received data replaces previously received data in receive RAM. Wraparound can simplify the interface with A/D converters by continuously updating conversion values stored in the RAM.

Continuous transfer mode allows simultaneous transfer of an uninterrupted bit stream. Any number of bits in a range from 8 to 256 can be transferred without CPU intervention. Longer transfers are possible, but minimal CPU intervention is required to prevent loss of data. A standard delay of 17 system clocks is inserted between each queue entry transfer.

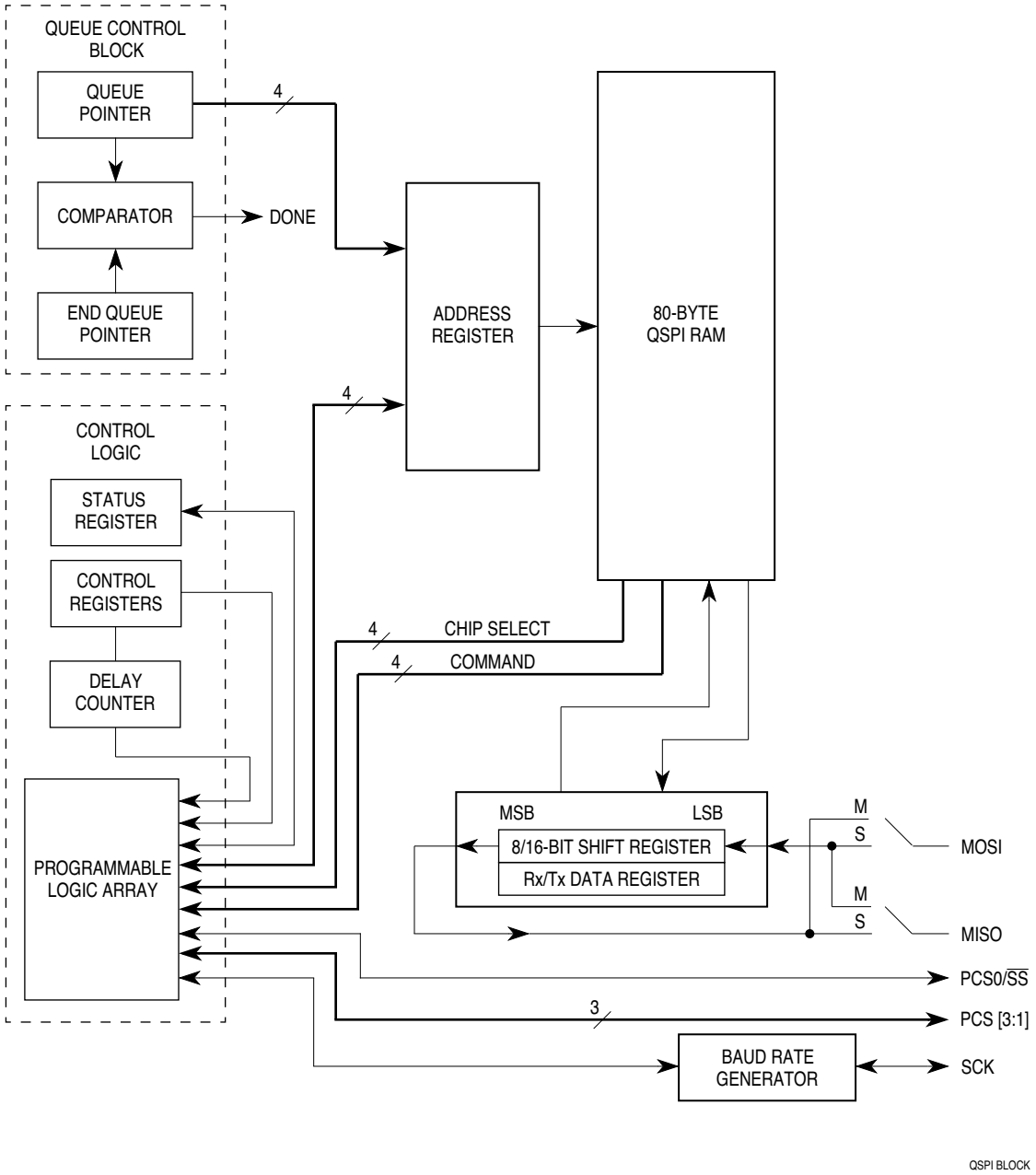


Figure 6-2 QSPI Block Diagram

6.3.1 QSPI Registers

The programmer's model for the QSPI consists of the QSM global and pin control registers, four QSPI control registers (SPCR[0:3]), a status register (SPCR), and the 80-byte QSPI RAM.

Registers and RAM can be read and written by the CPU. Refer to **APPENDIX D REGISTER SUMMARY** for register bit and field definitions.

6.3.1.1 Control Registers

Control registers contain parameters for configuring the QSPI and enabling various modes of operation. The CPU has read and write access to all control registers, but the QSM has read-only access to all bits except the SPE bit in SPCR1. Control registers must be initialized before the QSPI is enabled to ensure defined operation. SPCR1 must be written last because it contains the QSPI enable bit (SPE).

Writing a new value to any control register except SPCR2 while the QSPI is enabled disrupts operation. SPCR2 is buffered. New SPCR2 values become effective after completion of the current serial transfer. Rewriting NEWQP in SPCR2 causes execution to restart at the designated location. Reads of SPCR2 return the current value of the register, not of the buffer.

Writing the same value into any control register except SPCR2 while the QSPI is enabled has no effect on QSPI operation.

6.3.1.2 Status Register

The QSPI status register (SPSR) contains information concerning the current serial transmission. Only the QSPI can set the bits in this register. The CPU reads the SPSR to obtain QSPI status information and writes it to clear status flags.

6.3.2 QSPI RAM

The QSPI contains an 80-byte block of dual-access static RAM that can be accessed by both the QSPI and the CPU. The RAM is divided into three segments: receive data RAM, transmit data RAM, and command control data RAM. Receive data is information received from a serial device external to the MCU. Transmit data is information stored by the CPU for transmission to an external device. Command control data is used to perform transfers. Refer to **Figure 6-3**, which shows RAM organization.

6.3.2.1 Receive RAM

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 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.

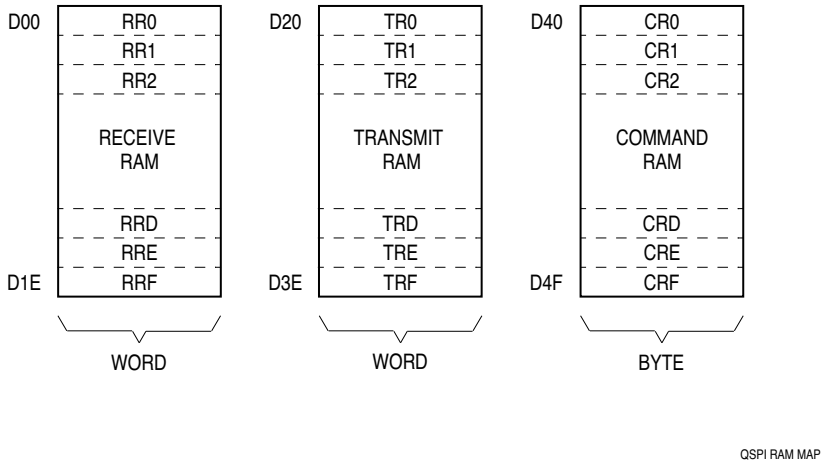


Figure 6-3 QSPI RAM

6.3.2.2 Transmit RAM

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

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

6.3.2.3 Command RAM

Command RAM is used by the QSPI 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 through the address in ENDQP (both of these fields are in SPCR2).

6.3.3 QSPI Pins

The QSPI uses seven pins. These pins can be configured for general-purpose I/O when not needed for QSPI application. When used for QSPI functions, the MOSI, MISO, and $\overline{SS}$ pins should have pull-up resistors.

Table 6-2 shows QSPI input and output pins and their functions.

Table 6-2 QSPI Pin Function

Pin/Signal Name	Mnemonic	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
Slave Select	$\overline{SS}$	Master Slave	Causes Mode Fault
Peripheral Chip Select 0	PCS0	Master	Initiates Serial Transfer Selects Peripherals

6.3.4 QSPI Operation

The QSPI uses a dedicated 80-byte block of static RAM accessible by both the QSPI and the CPU to perform queued operations. The RAM is divided into three segments. There are 16 command control bytes, 16 transmit data words, and 16 receive data words. QSPI RAM is organized so that one byte of command control data, one word of transmit data, and one word of receive data correspond to one queue entry, $0–$F.

The CPU initiates QSPI operation by setting up a queue of QSPI commands in command RAM, writing transmit data into transmit RAM, then enabling the QSPI. The QSPI executes the queued commands, sets a completion flag (SPIF), and then either interrupts the CPU or waits for CPU intervention.

There are four queue pointers. The CPU can access three of them through fields in QSPI registers. The new queue pointer (NEWQP), in SPCR2, points to the first command in the queue. An internal queue pointer points to the command currently being executed. The completed queue pointer (CPTQP), in SPSR, points to the last command executed. The end queue pointer (ENDQP), contained in SPCR2, points to the final command in the queue.

The internal pointer is initialized to the same value as NEWQP. During normal operation, the command pointed to by the internal pointer is executed, the value in the internal pointer is copied into CPTQP, the internal pointer is incremented, and then the sequence repeats. Execution continues at the internal pointer address unless the NEWQP value is changed. After each command is executed, ENDQP and CPTQP are compared. When a match occurs, the SPIF flag is set and the QSPI stops unless wrap-around mode is enabled.

At reset, NEWQP is initialized to $0. When the QSPI is enabled, execution begins at queue address $0 unless another value has been written into NEWQP. ENDQP is initialized to $0 at reset, but should be changed to show the last queue entry before the QSPI is enabled. NEWQP and ENDQP can be written at any time. When the NEWQP value changes, the internal pointer value also changes. However, if NEWQP is written while a transfer is in progress, the transfer is completed normally. Leaving NEWQP and ENDQP set to $0 causes a single transfer to occur when the QSPI is enabled.

6.3.5 QSPI 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 either mode is entered, appropriate QSM and QSPI registers must be initialized properly.

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 by the 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 multimaster 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.

Figure 6-4 shows QSPI initialization; **Figure 6-5** and **Figure 6-5** show QSPI master and slave operation. The CPU must initialize the QSM global and pin registers and the QSPI control registers before enabling the QSPI for either mode of operation (refer to **6.5 QSM Initialization**). The command queue must be written before the QSPI is enabled for master mode operation. Any data to be transmitted should be written into transmit RAM before the QSPI is enabled. During wraparound operation, data for subsequent transmissions can be written at any time.

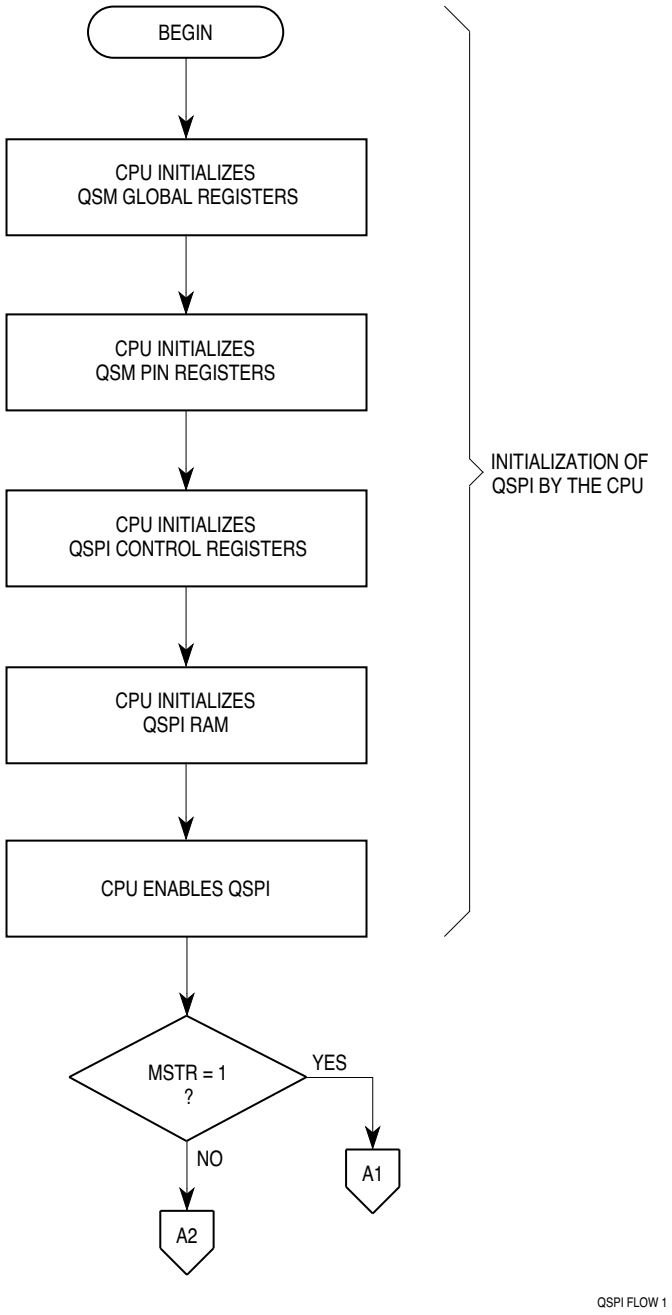


Figure 6-4 Flowchart of QSPI Initialization Operation

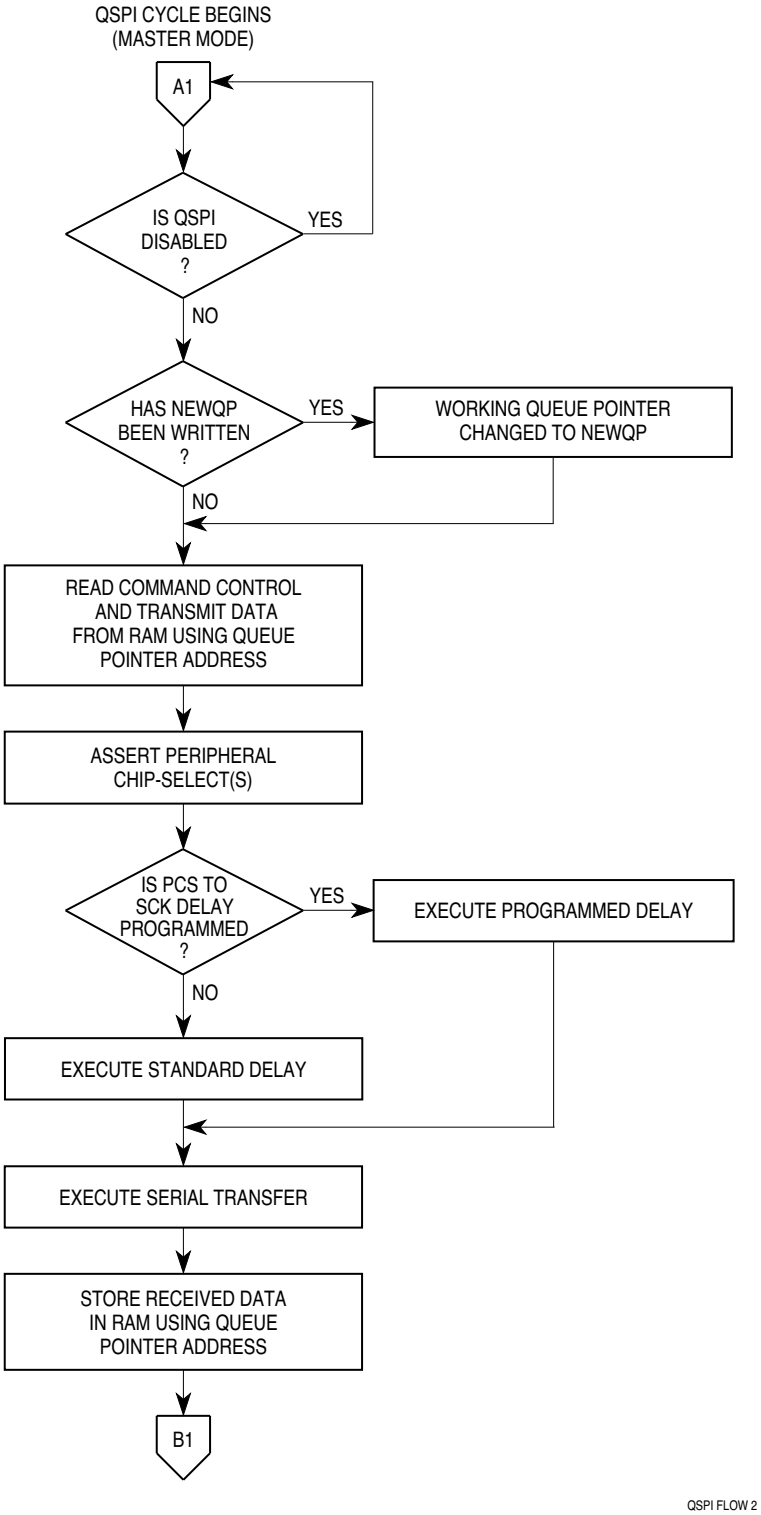


Figure 6-5 Flowchart of QSPI Master Operation (Part 1)

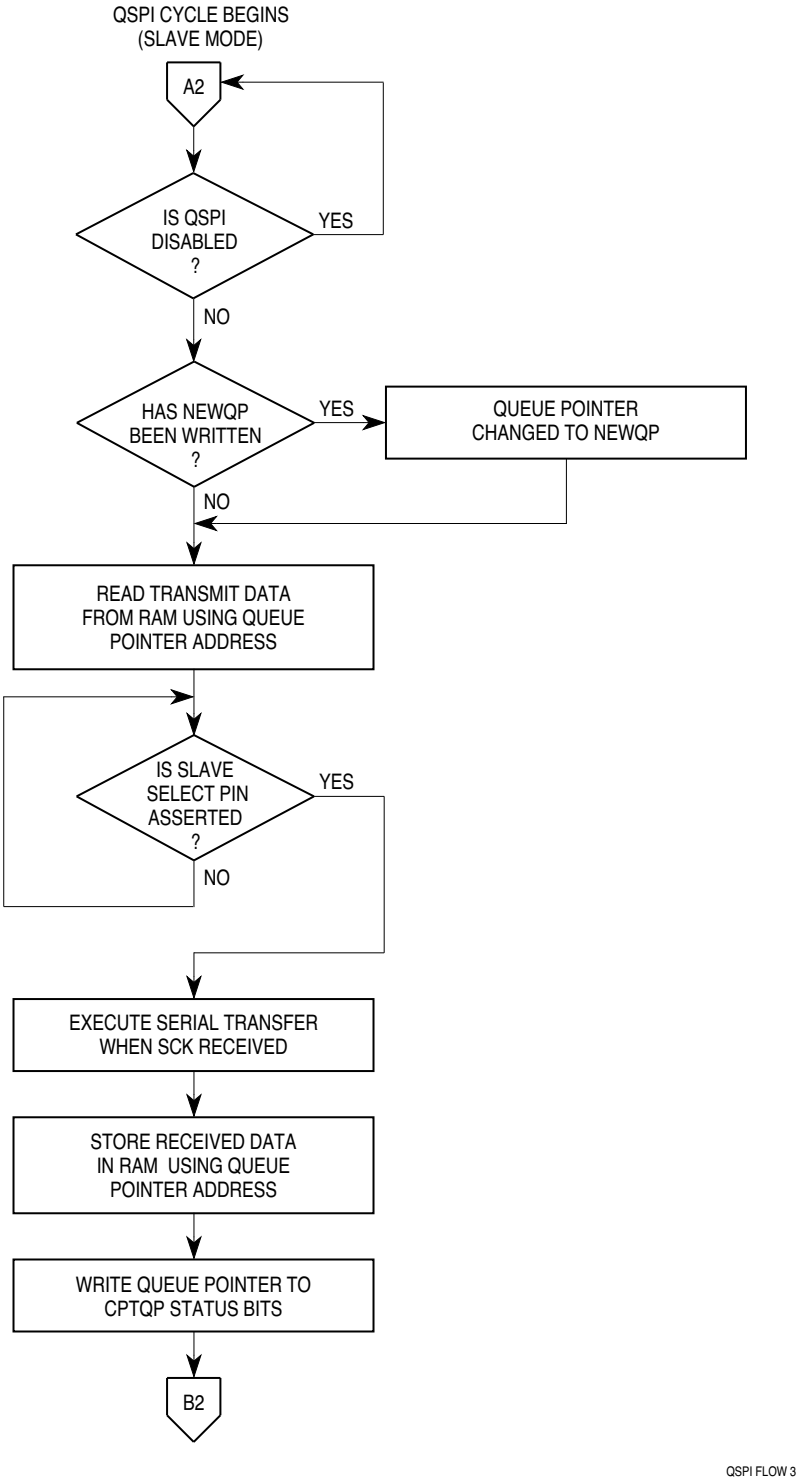


Figure 6-5 Flowchart of QSPI Master Operation (Part 2)

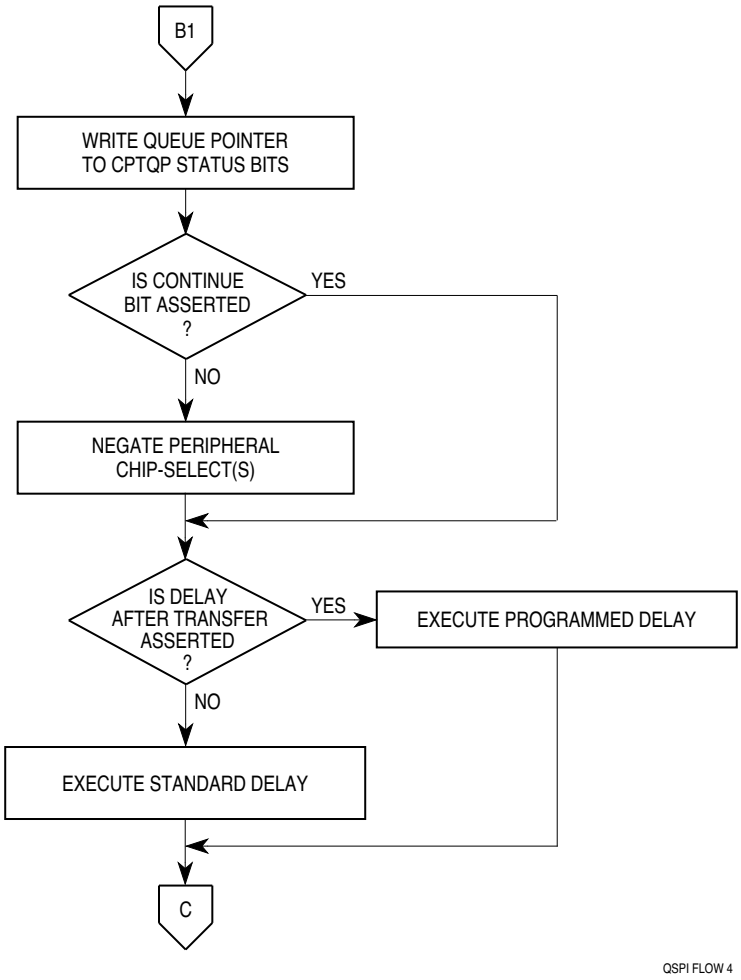


Figure 6-5 Flowchart of QSPI Master Operation (Part 3)

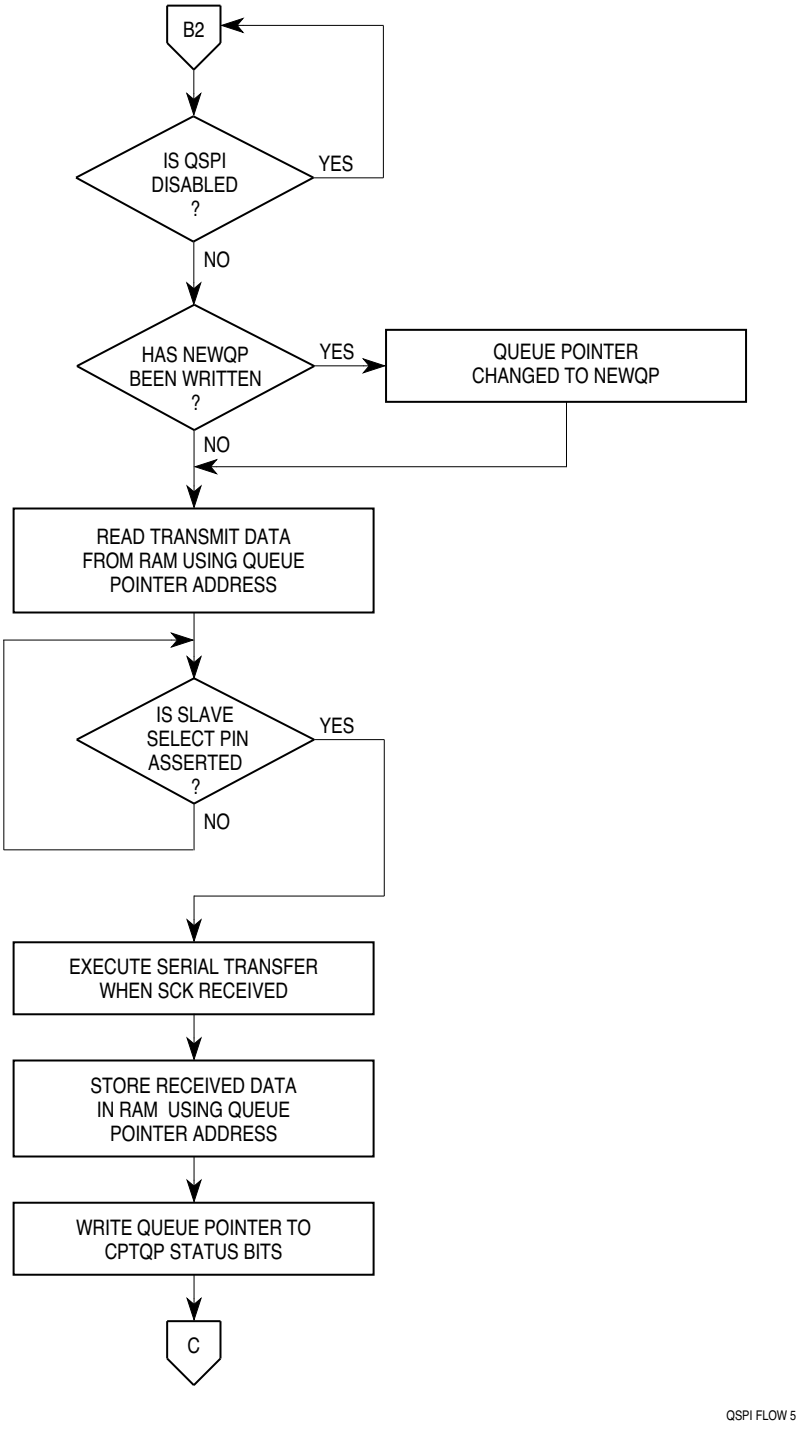


Figure 6-6 Flowchart of QSPI Slave Operation (Part 1)

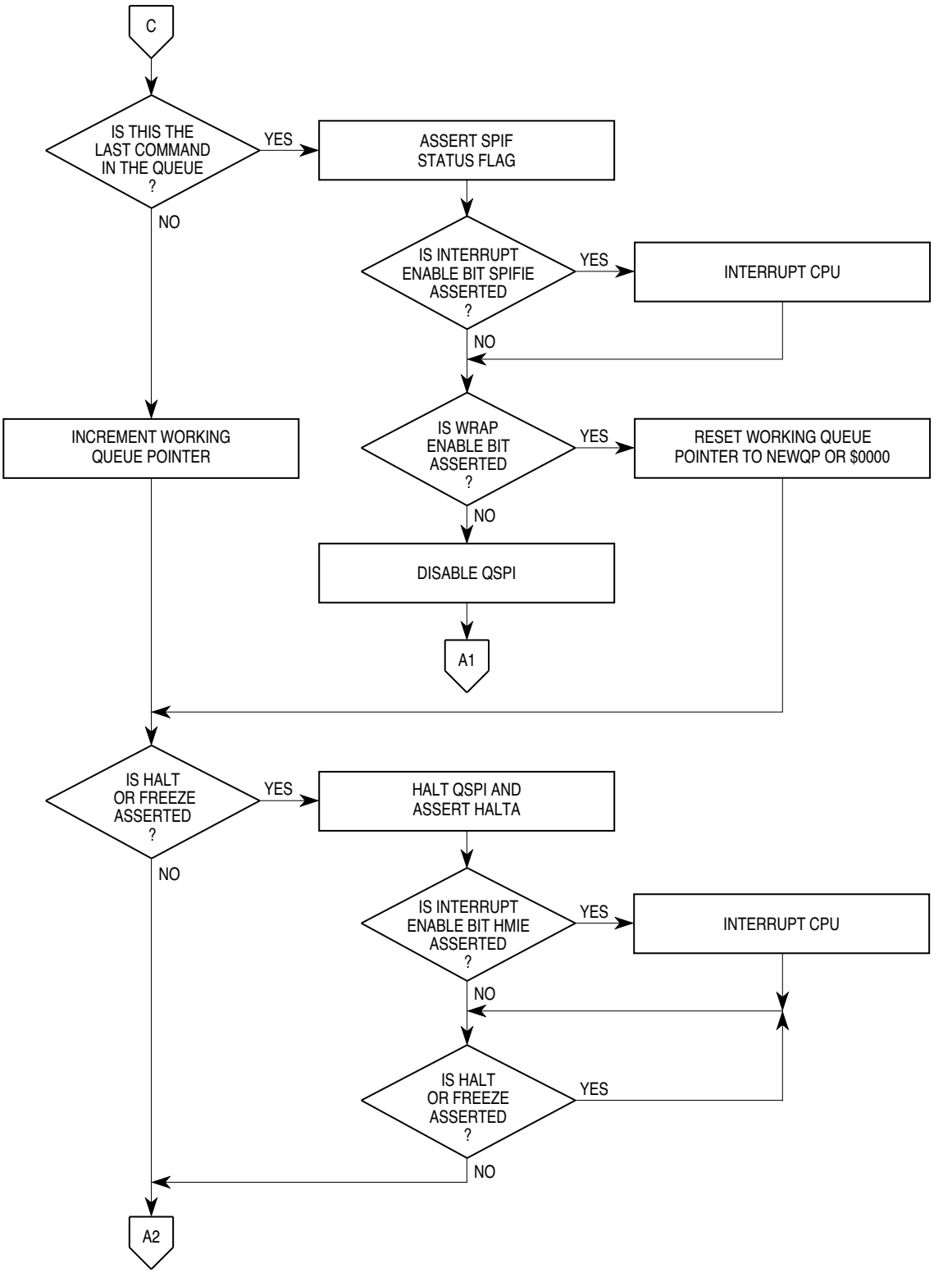


Figure 6-6 Flowchart of QSPI Slave Operation (Part 2)

Normally, the SPI bus performs synchronous bidirectional transfers. The serial clock on the SPI bus master supplies the clock signal (SCK) to time the transfer of data. Four possible combinations of clock phase and polarity can be specified by the CPHA and CPOL bits in SPCR0.

Data is transferred with the most significant bit first. The number of bits transferred per command defaults to eight, but can be set to any value from eight to sixteen bits by writing a value into the BITSE field in command RAM.

Typically, SPI bus outputs are not open-drain unless multiple SPI masters are in the system. If needed, the WOMQ bit in SPCR0 can be set to provide wired-OR, open-drain outputs. An external pull-up resistor should be used on each output line. WOMQ affects all QSPI pins regardless of whether they are assigned to the QSPI or used as general-purpose I/O.

6.3.5.1 Master Mode

Setting the MSTR bit in SPCR0 selects master mode operation. In master mode, the QSPI can initiate serial transfers, but cannot respond to externally initiated transfers. When the slave select input of a device configured for master mode is asserted, a mode fault occurs.

Before QSPI operation is initiated, QSM register PQSPAR must be written to assign necessary pins to the QSPI. The pins necessary for master mode operation are MISO and MOSI, SCK, and one or more of the chip-select pins. MISO is used for serial data input in master mode, and MOSI is used for serial data output. Either or both may be necessary, depending on the particular application. SCK is the serial clock output in master mode.

Before master mode operation is initiated, QSM register DDRQS must be written to direct the data flow on the QSPI pins used. Configure the SCK, MOSI and appropriate chip-select pins PCS[3:0]/$\overline{SS}$ as outputs. The MISO pin must be configured as an input.

After pins are assigned and configured, write appropriate data to the command queue. If data is to be transmitted, write the data to transmit RAM. Initialize the queue pointers as appropriate.

Data transfer is synchronized with the internally-generated serial clock (SCK). Control bits, CPHA and CPOL, in SPCR0, control clock phase and polarity. Combinations of CPHA and CPOL determine upon which SCK edge to drive outgoing data from the MOSI pin and to latch incoming data from the MISO pin.

Baud rate is selected by writing a value from 2 to 255 into the SPBR field in SPCR0. The QSPI uses a modulus counter to derive SCK baud rate from the MCU system clock.

The following expressions apply to SCK baud rate:

$$\text{SCK Baud Rate} = \frac{\text{System Clock}}{2 \times \text{SPBR}}$$

or

$$\text{SPBR} = \frac{\text{System Clock}}{(2 \times \text{SCK})(\text{Baud Rate Desired})}$$

Giving SPBR a value of zero or one disables the baud rate generator. SCK is disabled and assumes its inactive state value.

The DSCK field in command RAM determines the delay period from chip-select assertion until the leading edge of the serial clock. The DSCKL field in SPCR1 determines the period of delay before the assertion of SCK. The following expression determines the actual delay before SCK:

$$\text{PCS to SCK Delay} = \frac{\text{DSCKL}}{\text{System Clock Frequency}}$$

where DSCKL equals {1, 2, 3,..., 127}.

When DSCK equals zero, DSCKL is not used. Instead, the PCS valid-to-SCK transition is one-half the DSCK period.

There are two transfer length options. The user can choose a default value of eight bits, or a programmed value of eight to sixteen bits, inclusive. The programmed value must be written into the BITS field in SPCR0. The BITSE field in command RAM determines whether the default value (BITSE = 0) or the BITS value (BITSE = 1) is used. **Table 6-3** shows BITS field encoding.

Table 6-3 BITS Encoding

BITS	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

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. There are two transfer delay options. The user can choose to delay a standard period after serial transfer is complete or can specify a delay period. Writing a value to the DTL field in SPCR1 specifies a delay period. The DT bit in command RAM determines whether the standard delay period (DT = 0) or the specified delay period (DT = 1) is used. The following expression is used to calculate the delay:

$$\text{Delay after Transfer} = \frac{32 \times \text{DTL}}{\text{System Clock Frequency}}$$

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

A zero value for DTL causes a delay-after-transfer value of 8192/system clock.

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

Adequate delay between transfers must be specified for long data streams because the QSPI requires time to load a transmit RAM entry for transfer. Receiving devices need at least the standard delay between successive transfers. If the system clock is operating at a slower rate, the delay between transfers must be increased proportionately.

Operation is initiated by setting the SPE bit in SPCR1. Shortly after SPE is set, the QSPI executes the command at the command RAM address pointed to by NEWQP. Data at the pointer address in transmit RAM is loaded into the data serializer and transmitted. Data that is simultaneously received is stored at the pointer address in receive RAM.

When the proper number of bits have been transferred, the QSPI stores the working queue pointer value in CPTQP, increments the working queue pointer, and loads the next data for transfer from transmit RAM. The command pointed to by the incremented working queue pointer is executed next, unless a new value has been written to NEWQP. If a new queue pointer value is written while a transfer is in progress, that transfer is completed normally.

When the CONT bit in command RAM is set, PCS pins are continuously driven in specified states during and between transfers. If the chip-select pattern changes during or between transfers, the original pattern is driven until execution of the following transfer begins. When CONT is cleared, the data in register PORTQS is driven between transfers.

When the QSPI reaches the end of the queue, it sets the SPIF flag. If the SPIFIE bit in SPCR2 is set, an interrupt request is generated when SPIF is asserted. At this point, the QSPI clears SPE and stops unless wraparound mode is enabled.

6.3.5.2 Master Wraparound Mode

Wraparound mode is enabled by setting the WREN bit in SPCR2. The queue can wrap to pointer address $0 or to the address pointed to by NEWQP, depending on the state of the WRTO bit in SPCR2.

In wraparound mode, the QSPI cycles through the queue continuously, even while the QSPI is requesting interrupt service. SPE is not cleared when the last command in the queue is executed. New receive data overwrites previously received data in receive RAM. Each time the end of the queue is reached, the SPIF flag is set. SPIF is not automatically reset. If interrupt-driven SPI service is used, the service routine must clear the SPIF bit to abort the current request. Additional interrupt requests during servicing can be prevented by clearing SPIFIE, but SPIFIE is buffered. Clearing it does not abort a current request.

There are two recommended methods of exiting wraparound mode: clearing the WREN bit or setting the HALT bit in SPCR3. Exiting wraparound mode by clearing SPE is not recommended, as clearing SPE may abort a serial transfer in progress. The QSPI sets SPIF, clears SPE, and stops the first time it reaches the end of the queue after WREN is cleared. After HALT is set, the QSPI finishes the current transfer, then stops executing commands. After the QSPI stops, SPE can be cleared.

6.3.5.3 Slave Mode

Clearing the MSTR bit in SPCR0 selects slave mode operation. In slave mode, the QSPI is unable to initiate serial transfers. Transfers are initiated by an external bus master. Slave mode is typically used on a multi-master SPI bus. Only one device can be bus master (operate in master mode) at any given time.

Before QSPI operation is initiated, QSM register PQSPAR must be written to assign necessary pins to the QSPI. The pins necessary for slave mode operation are MISO and MOSI, SCK, and PCS0/$\overline{SS}$. MISO is used for serial data output in slave mode, and MOSI is used for serial data input. Either or both may be necessary, depending on the

particular application. SCK is the serial clock input in slave mode. Assertion of the active-low slave select signal $\overline{SS}$ initiates slave mode operation.

Before slave mode operation is initiated, DDRQS must be written to direct data flow on the QSPI pins used. Configure the MOSI, SCK and PCS0/$\overline{SS}$ pins as inputs. The MISO pin must be configured as an output.

After pins are assigned and configured, write data to be transmitted into transmit RAM. Command RAM is not used in slave mode and does not need to be initialized. Unused portions of QSPI RAM can be used by the CPU as general-purpose RAM. Initialize the queue pointers as appropriate.

When SPE is set and MSTR is clear, a low state on the slave select (PCS0/$\overline{SS}$) pin begins slave mode operation at the address indicated by NEWQP. Data that is received is stored at the pointer address in receive RAM. Data is simultaneously loaded into the data serializer from the pointer address in transmit RAM and transmitted. Transfer is synchronized with the externally generated SCK. The CPHA and CPOL bits determine on which SCK edge to latch incoming data from the MISO pin and to drive outgoing data from the MOSI pin.

Because the command control segment is not used, the command control bits and peripheral chip-select codes have no effect in slave mode operation. The PCS0/$\overline{SS}$ pin is used only as an input.

The SPBR, DT and DSCK bits are not used in slave mode. The QSPI drives neither the clock nor the chip-select pins and thus cannot control clock rate or transfer delay.

Because the BITSE option is not available in slave mode, the BITS field specifies the number of bits to be transferred for all transfers in the queue. When the number of bits designated by BITS has been transferred, the QSPI stores the working queue pointer value in CPTQP, increments the working queue pointer, and loads new transmit data from transmit RAM into the data serializer. The working queue pointer address is used the next time PCS0/$\overline{SS}$ is asserted, unless the CPU writes to NEWQP first.

The QSPI shifts one bit for each pulse of SCK until the slave select input goes high. If $\overline{SS}$ goes high before the number of bits specified by the BITS field is transferred, the QSPI resumes operation at the same pointer address the next time $\overline{SS}$ is asserted. The maximum value that the BITS field can have is 16. If more than 16 bits are transmitted before $\overline{SS}$ is negated, pointers are incremented and operation continues. The QSPI transmits as many bits as it receives at each queue address, until the BITS value is reached or $\overline{SS}$ is negated. $\overline{SS}$ does not need to go high between transfers, as the QSPI transfers data until reaching the end of the queue, whether $\overline{SS}$ remains low or is toggled between transfers.

When the QSPI reaches the end of the queue, it sets the SPIF flag. If the SPIFIE bit in SPCR2 is set, an interrupt request is generated when SPIF is asserted. At this point, the QSPI clears SPE and stops unless wraparound mode is enabled.

6.3.5.4 Slave Wraparound Mode

Slave wraparound mode is enabled by setting the WREN bit in SPCR2. The queue can wrap to pointer address $0 or to the address pointed to by NEWQP, depending on the state of the WRTO bit in SPCR2. Slave wraparound operation is identical to master wraparound operation.

6.3.6 Peripheral Chip Selects

Peripheral chip-select signals are used to select an external device for serial data transfer. Chip-select signals are asserted when a command in the queue is executed. Signals are asserted at a logic level corresponding to the value of the PCS bits in the command. More than one chip-select signal can be asserted at a time, and more than one external device can be connected to each PCS pin, provided proper fanout is observed. PCS0 shares a pin with the slave select $\overline{SS}$ signal, which initiates slave mode serial transfer. If $\overline{SS}$ is taken low when the QSPI is in master mode, a mode fault occurs.

To set up a chip-select function, set the appropriate bit in PQSPAR, then configure the chip-select pin as an output by setting the appropriate bit in DDRQS. The value of the bit in PORTQS that corresponds to the chip-select pin determines the base state of the chip-select signal. If base state is zero, chip-select assertion must be active high (PCS bit in command RAM must be set); if base state is one, assertion must be active low (PCS bit in command RAM must be cleared). PORTQS bits are cleared during reset. If no new data is written to PORTQS before pin assignment and configuration as an output, base state of chip-select signals is zero and chip-select pins are configured for active-high operation.

6.4 Serial Communication Interface

The serial communication interface (SCI) communicates with external devices through an asynchronous serial bus. The SCI uses a standard nonreturn to zero (NRZ) transmission format. The SCI is fully compatible with other Freescale SCI systems, such as those in M68HC11 and M68HC05 devices. **Figure 6-7** is a block diagram of the SCI transmitter; **Figure 6-8** is a block diagram of the SCI receiver.

6.4.1 SCI Registers

The SCI programming model includes the QSM global and pin control registers, and four SCI registers. There are two SCI control registers (SCCR0 and SCCR1), one status register (SCSR), and one data register (SCDR). Refer to **APPENDIX D REGISTER SUMMARY** for register bit and field definition.

6.4.1.1 Control Registers

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

SCCR1 contains a number of SCI configuration parameters, including transmitter and receiver enable bits, interrupt enable bits, and operating mode enable bits. The CPU can read and write this register at any time. The SCI can modify the RWU bit under certain circumstances.

Changing the value of SCI control bits during a transfer operation may disrupt operation. Before changing register values, allow the SCI to complete the current transfer, then disable the receiver and transmitter.

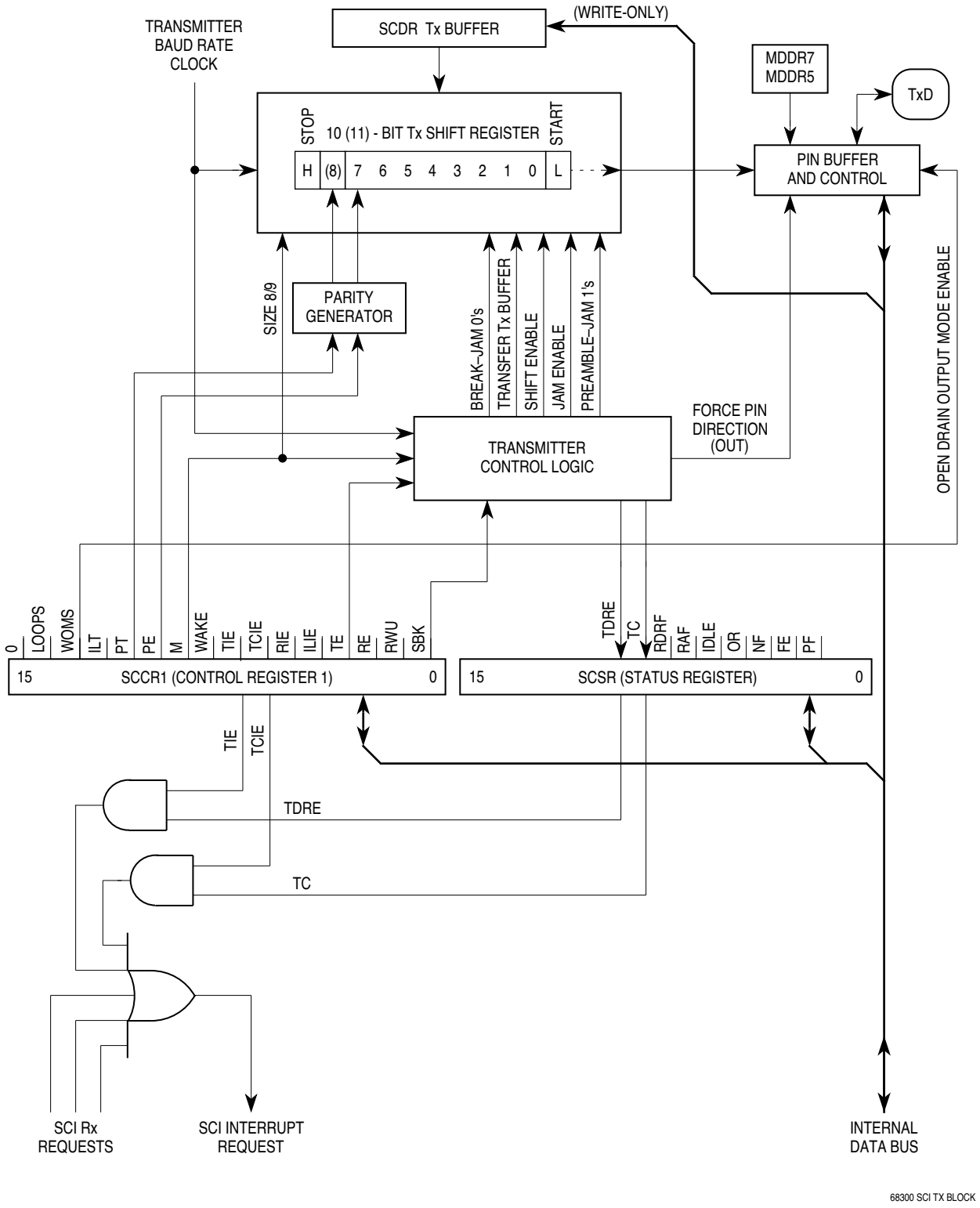


Figure 6-7 SCI Transmitter Block Diagram

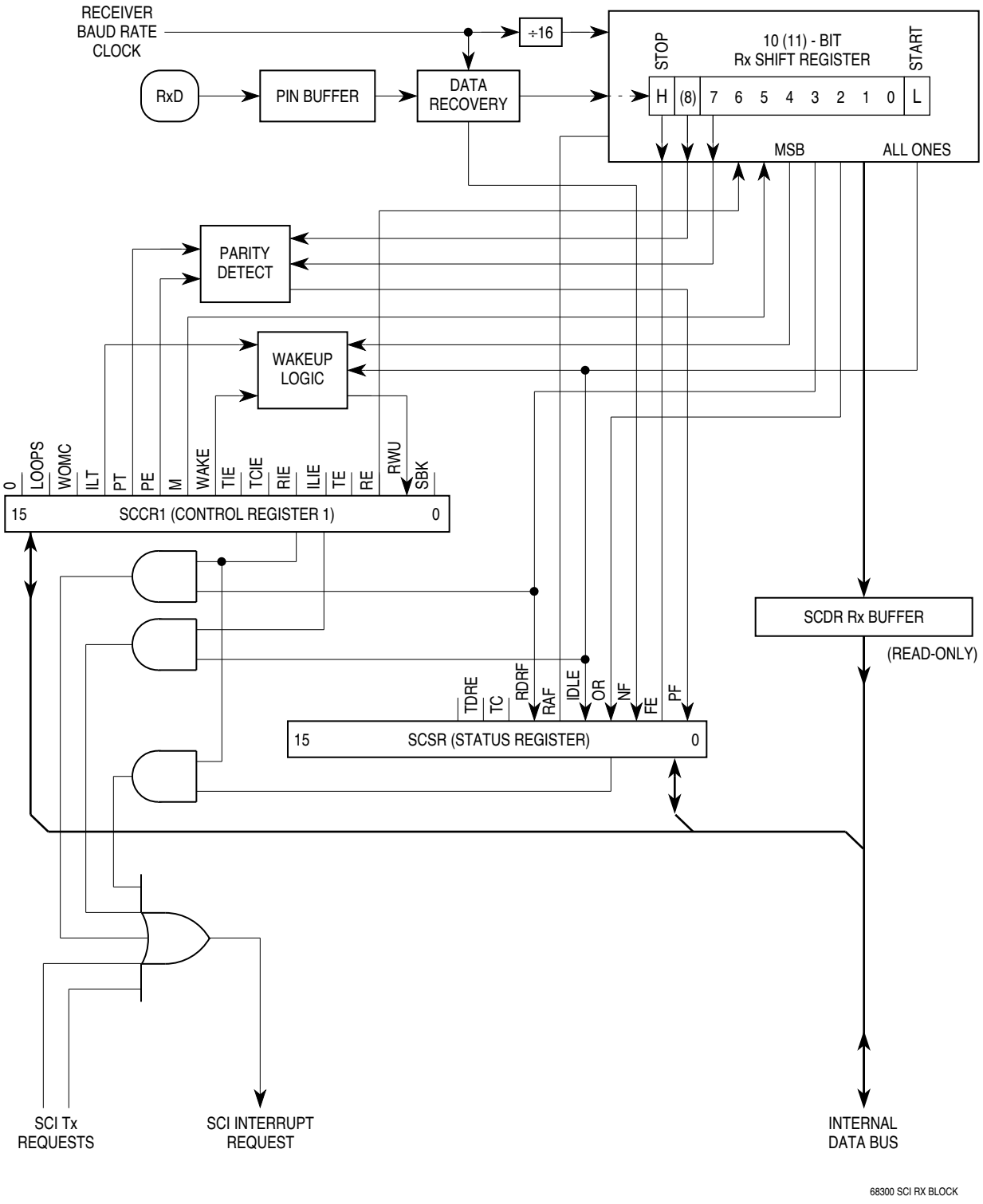


Figure 6-8 SCI Receiver Block Diagram

6.4.1.2 Status Register

The SCI status register (SCSR) contains flags that show SCI operating conditions. These flags are cleared either by SCI hardware or by a read/write sequence. In general, flags are cleared by reading the SCSR, then reading (receiver status bits) or writing (transmitter status bits) the SCDR. A long-word read can consecutively access both the 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 the SCDR, the newly set status bit is not cleared. The SCSR must be read again with the bit set, and the SCDR must be written or read before the status bit is cleared.

Reading either byte of the SCSR causes all 16 bits to be accessed, and any status bit already set in either byte is cleared on a subsequent read or write of the SCDR.

6.4.1.3 Data Register

The SCDR contains two data registers at the same address. The RDR is a read-only register that contains data received by the SCI serial interface. The data comes into the receive serial shifter and is transferred to the RDR. The TDR is a write-only register that contains data to be transmitted. The data is first written to the TDR, 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 the SCDR is read, or the first eight data bits to be transmitted when the 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.

6.4.2 SCI Pins

Two unidirectional pins, TXD (transmit data) and RXD (receive data), are associated with the SCI. TXD can be used by the SCI or for general-purpose I/O. Function is assigned by the port QS pin assignment register (PQSPAR). The receive data (RXD) pin is dedicated to the SCI. **Table 6-4** shows SCI pin function.

Table 6-4 SCI Pin Function

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-Purposed I/O Serial Data Output from SCI

6.4.3 SCI Operation

SCI status flags in the SPSR support polled operation, or interrupt-driven operation can be employed by the interrupt enable bits in SCCR1.

6.4.3.1 Definition of Terms

- Bit-Time — The time required to transmit or receive one bit of data; one cycle of the baud frequency.

- Start Bit — One bit-time of logic zero that indicates the beginning of a data frame. A start bit must begin with a one-to-zero transition and be preceded by at least three receive time (RT) samples of logic one.
- Stop Bit — One bit-time of logic one that indicates the end of a data frame.
- Frame — A complete unit of serial information. The SCI can use 10-bit or 11-bit frames.
- Data Frame — A start bit, a specified number of data or information bits, and at least one stop bit.
- Idle Frame — A frame that consists of consecutive ones. An idle frame has no start bit.
- Break Frame — A frame that consists of consecutive zeros. A break frame has no stop bits.

6.4.3.2 Serial Formats

All data frames must have a start bit and at least one stop bit. Receiving and transmitting devices must use the same data frame format. The SCI provides hardware support for both ten-bit and eleven-bit frames. The serial mode (M) bit in SCI control register one (SCCR1) specifies the number of bits per frame.

The most common ten-bit data frame format for NRZ serial interface consists of one start bit, eight data bits (LSB first), and one stop bit. The most common eleven-bit data frame contains one start bit, eight data bits, a parity or control bit, and one stop bit. Ten-bit and eleven-bit frames are shown in **Table 6-5**.

Table 6-5 Serial Frame Formats

10-Bit Frames			
Start	**Data**	**Parity/Control**	**Stop**
1	7	—	2
1	7	1	1
1	8	—	1
11-Bit Frames			
Start	**Data**	**Parity/Control**	**Stop**
1	7	1	2
1	8	1	1

6.4.3.3 Baud Clock

The SCI baud clock is programmed by writing a 13-bit value to the baud rate (SCBR) field in SCI control register zero (SCCR0). Baud clock is derived from the MCU system clock by a modulus counter. Writing a value of zero to SCBR disables the baud rate generator. Baud clock rate is calculated as follows:

$$\text{SCI Baud Clock Rate} = \frac{\text{System Clock}}{32 \times \text{SCBR}}$$

where SCBR is in the range {1, 2, 3,..., 8191}.

The SCI receiver operates asynchronously. An internal clock is necessary to synchronize with an incoming data stream. The SCI baud clock generator produces a receive

time (RT) sampling clock with a frequency 16 times that of the SCI baud clock. 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.

6.4.3.4 Parity Checking

The parity type (PT) bit in SCCR1 selects either even (PT = 0) or odd (PT = 1) parity. PT affects received and transmitted data. The parity enable (PE) bit in SCCR1 determines whether parity checking is enabled (PE = 1) or disabled (PE = 0). When PE is set, the MSB of the data in a frame is used for the parity function. For transmitted data, a parity bit is generated; for received data, the parity bit is checked. When parity checking is enabled, the parity flag (PF) in the SCI status register (SCSR) is set if a parity error is detected.

Enabling parity affects the number of data bits in a frame, which can in turn affect frame size. **Table 6-6** shows possible data and parity formats.

Table 6-6 Effect of Parity Checking on Data Size

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

6.4.3.5 Transmitter Operation

The transmitter consists of a serial shifter and a parallel data register (TDR) located in the SCI data register (SCDR). The serial shifter cannot be directly accessed by the CPU. The transmitter is double-buffered, which means that data can be loaded into the TDR while other data is shifted out. The transmitter enable (TE) bit in SCCR1 enables (TE = 1) and disables (TE = 0) the transmitter.

Shifter output is connected to the TXD pin while the transmitter is operating (TE = 1, or TE = 0 and transmission in progress). Wired-OR operation should be specified when more than one transmitter is used on the same SCI bus. The wired-OR mode select bit (WOMS) in SCCR1 determines whether TXD is an open-drain (wired-OR) output or a normal CMOS output. An external pull-up resistor on the TXD pin is necessary for wired-OR operation. WOMS controls TXD function whether the pin is used for SCI transmissions (TE = 1) or as a general-purpose I/O pin.

Data to be transmitted is written to TDR, then transferred to the serial shifter. The transmit data register empty (TDRE) flag in SCSR shows the status of TDR. When TDRE = 0, TDR contains data that has not been transferred to the shifter. Writing to TDR again overwrites the data. TDRE is set when the data in TDR is transferred to the shifter. Before new data can be written to TDR, however, the processor must clear TDRE by writing to SCSR. If new data is written to TDR without first clearing TDRE, the data will not be transmitted.

The transmission complete (TC) flag in SCSR shows transmitter shifter state. When TC = 0, the shifter is busy. TC is set when all shifting operations are completed. TC is not automatically cleared. The processor must clear it by first reading SCSR while TC is set, then writing new data to TDR.

The state of the serial shifter is checked when the TE bit is set. If TC = 1, an idle frame is transmitted as a preamble to the following data frame. If TC = 0, the current operation continues until the final bit in the frame is sent, then the preamble is transmitted. The TC bit is set at the end of preamble transmission.

The send break (SBK) bit in SCCR1 is used to insert break frames in a transmission. A nonzero integer number of break frames is transmitted while SBK is set. Break transmission begins when SBK is set, and ends with the transmission in progress at the time either SBK or TE are cleared. If SBK is set while a transmission is in progress, that transmission finishes normally before the break begins. To assure the minimum break time, toggle SBK quickly to one and back to zero. The TC bit is set at the end of break transmission. After break transmission, at least one bit-time of logic level one (mark idle) is transmitted to ensure that a subsequent start bit can be detected.

If TE remains set, after all pending idle, data and break frames are shifted out, TDRE and TC are set and TXD is held at logic level one (mark).

When TE is cleared, the transmitter is disabled after all pending idle, data and break frames are transmitted. The TC flag is set, and the TXD pin reverts to control by PQSPAR and DDRQS. Buffered data is not transmitted after TE is cleared. To avoid losing data in the buffer, do not clear TE until TDRE is set.

Some serial communication systems require a mark on the TXD pin even when the transmitter is disabled. Configure the TXD pin as an output (DDRQS), then write a one to PORTQS bit 7. When the transmitter releases control of the TXD pin, it reverts to driving a logic one output.

To insert a delimiter between two messages, to place nonlistening receivers in wakeup mode between transmissions, or to signal a retransmission by forcing an idle line, clear and then set TE before data in the serial shifter has shifted out. The transmitter finishes the transmission, then sends a preamble. After the preamble is transmitted, if TDRE is set, the transmitter will mark idle. Otherwise, normal transmission of the next sequence will begin.

Both TDRE and TC have associated interrupts. The interrupts are enabled by the transmit interrupt enable (TIE) and transmission complete interrupt enable (TCIE) bits in SCCR1. Service routines can load the last byte of data in a sequence into the TDR, then terminate the transmission when a TDRE interrupt occurs.

6.4.3.6 Receiver Operation

The receiver enable (RE) bit in SCCR1 enables (RE = 1) and disables (RE = 0) the transmitter. The receiver contains a receive serial shifter and a parallel receive data register (RDR) located in the SCI data register (SCDR). The serial shifter cannot be directly accessed by the CPU. The receiver is double-buffered, allowing data to be held in RDR while other data is shifted in.

Receiver bit processor logic drives a state machine that determines the logic level for each bit-time. This state machine controls when the bit processor logic is to sample the RXD pin and also controls when data is to be passed to the receive serial shifter. A receive time (RT) clock is used to control sampling and synchronization. Data is shifted into the receive serial shifter according to the most recent synchronization of the RT clock with the incoming data stream. From this point on, data movement is synchronized with the MCU system clock. Operation of the receiver state machine is detailed in the *QSM Reference Manual* (QSMRM/AD).

The number of bits shifted in by the receiver depends on the serial format. However, all frames must end with at least one stop bit. When the stop bit is received, the frame is considered to be complete, and the received data in the serial shifter is transferred to the RDR. The receiver data register flag (RDRF) is set when the data is transferred.

Noise errors, parity errors, and framing errors can be detected while a data stream is being received. Although error conditions are detected as bits are received, the noise flag (NF), the parity flag (PF), and the framing error (FE) flag in SCSR are not set until data is transferred from the serial shifter to RDR.

RDRF must be cleared before the next transfer from the shifter can take place. If RDRF is set when the shifter is full, transfers are inhibited and the overrun error (OR) flag in the SCSR is set. OR indicates that the CPU needs to service RDR faster. When OR is set, the data in RDR is preserved, but the data in the serial shifter is lost. Because framing, noise, and parity errors are detected while data is in the serial shifter, FE, NF, and PF cannot occur at the same time as OR.

When the CPU reads the SCSR and the SCDR in sequence, it acquires status and data, and also clears the status flags. Reading the SCSR acquires status and arms the clearing mechanism. Reading the SCDR acquires data and clears the SCSR.

When RIE in SCCR1 is set, an interrupt request is generated whenever RDRF is set. Because receiver status flags are set at the same time as RDRF, they do not have separate interrupt enables.

6.4.3.7 Idle-Line Detection

During a typical serial transmission, frames are transmitted isochronously and no idle time occurs between frames. Even when all the data bits in a frame are logic ones, the start bit provides one logic zero bit-time during the frame. An idle line is a sequence of contiguous ones equal to the current frame size. Frame size is determined by the state of the M bit in SCCR1.

The SCI receiver has both short and long idle-line detection capability. Idle-line detection is always enabled. The idle line type (ILT) bit in SCCR1 determines which type of detection is used. When an idle line condition is detected, the IDLE flag in SCSR is set.

For short idle-line detection, the receiver bit processor counts contiguous logic one bit-times whenever they occur. Short detection provides the earliest possible recognition of an idle line condition, because the stop bit and contiguous logic ones before and after it are counted. For long idle-line detection, the receiver counts logic ones after the stop bit is received. Only a complete idle frame causes the IDLE flag to be set.

In some applications, CPU overhead can cause a bit-time of logic level one to occur between frames. This bit-time does not affect content, but if it occurs after a frame of ones when short detection is enabled, the receiver flags an idle line.

When the idle line interrupt enable (ILIE) bit in SCCR1 is set, an interrupt request is generated when the IDLE flag is set. The flag is cleared by reading SCSR and SCDR in sequence. IDLE is not set again until after at least one frame has been received (RDRF = 1). This prevents an extended idle interval from causing more than one interrupt.

6.4.3.8 Receiver Wakeup

The receiver wakeup function allows a transmitting device to direct a transmission to a single receiver or to a group of receivers by sending an address frame at the start of a message. Hardware activates each receiver in a system under certain conditions. Resident software must process address information and enable or disable receiver operation.

A receiver is placed in wakeup mode by setting the receiver wakeup (RWU) bit in SCCR1. While RWU is set, receiver status flags and interrupts are disabled. Although the CPU can clear RWU, it is normally cleared by hardware during wakeup.

The WAKE bit in SCCR1 determines which type of wakeup is used. When WAKE = 0, idle-line wakeup is selected. When WAKE = 1, address-mark wakeup is selected. Both types require a software-based device addressing and recognition scheme.

Idle-line wakeup allows a receiver to sleep until an idle line is detected. When an idle-line is detected, the receiver clears RWU and wakes up. The receiver waits for the first frame of the next transmission. The byte is received normally, transferred to register RDR, and the RDRF flag is set. If software does not recognize the address, it can set RWU and put the receiver back to sleep. For idle-line wakeup to work, there must be a minimum of one frame of idle line between transmissions. There must be no idle time between frames within a transmission.

Address-mark wakeup uses a special frame format to wake up the receiver. When the MSB of an address-mark frame is set, that frame contains address information. The first frame of each transmission must be an address frame. When the MSB of a frame is set, the receiver clears RWU and wakes up. The byte is received normally, transferred to register RDR, and the RDRF flag is set. If software does not recognize the address, it can set RWU and put the receiver back to sleep. Address-mark wakeup allows idle time between frames and eliminates idle time between transmissions. However, there is a loss of efficiency because of an additional bit-time per frame.

6.4.3.9 Internal Loop

The LOOPS bit in SCCR1 controls a feedback path on the data serial shifter. When LOOPS is set, 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.

6.5 QSM Initialization

After reset, the QSM remains in an idle state until initialized. A general sequence guide for initialization follows.

A. Global
   1. Configuration register (QSMCR)
      a. Write an interrupt arbitration priority value into the IARB field.
      b. Clear the FREEZE and/or STOP bits for normal operation.
   2. Interrupt vector and interrupt level registers (QIVR and QILR)
      a. Write QSPI/SCI interrupt vector into QIVR.
      b. Write QSPI (ILSPI) and SCI (ILSCI) interrupt priorities into QILR.
   3. Port data and data direction registers (PORTQS and DDRQS)
      a. Write a data word to PORTQS.
      b. Establish direction of QSM pins used for I/O by writing to DDRQS.
   4. Assign pin functions by writing to the pin assignment register (PQSPAR)

B. Queued Serial Peripheral Interface
   1. Write appropriate values to QSPI command RAM.
   2. QSPI control register zero (SPCR0)
      a. Write a transfer rate value into the BR field.
      b. Determine clock phase (CPHA), and clock polarity (CPOL).
      c. Determine number of bits to be transferred in a serial operation (BIT).
      d. Select master or slave operating mode (MSTR).
      e. Enable or disable wired-OR operation (WOMQ).
   3. QSPI control register one (SPCR1)
      a. Establish a delay following serial transfer by writing to the DTL field.
      b. Establish a delay before serial transfer by writing to the DSCKL field.
   4. QSPI control register two (SPCR2)
      a. Write an initial queue pointer value into the NEWQP field.
      b. Write a final queue pointer value into the ENDQP field.
      c. Enable or disable queue wraparound (WREN).
      d. Write wraparound address into the WRTO field.
      e. Enable or disable QSPI flag interrupt (SPIFIE).
   5. QSPI control register three (SPCR3)
      a. Enable or disable halt at end of queue (HALT).
      b. Enable or disable halt and mode fault interrupts (HMIE).
      c. Enable or disable loopback (LOOPQ).
   6. To enable the QSPI, set the SPE bit in SPCR1.

C. Serial Communication Interface (SCI)
   1. SCI control register zero (SCCR0)
      a. Write a transfer rate (baud) value into the BR field.
   2. SCI control register one (SCCR1)
      a. Select serial mode (M)
      b. Enable use (PE) and type (PT) of parity check.
      c. Select use (RWU) and type (WAKE) of receiver wakeup.
      d. Enable idle-line detection (ILT) and interrupt (ILIE).
      e. Enable or disable wired-OR operation (WOMS).
      f. Enable or disable break transmission (BK).

3. To receive
   a. Set the receiver (RE) and receiver interrupt (RIE) bits in SCCR1.
4. To transmit
   a. Set transmitter (TE) and transmitter interrupt (TIE).
   b. Clear the transmitter data register empty (TDRE) and transmit complete (TC) indicators by reading the serial communication interface status register (SCSR).
   c. Write transmit data to the serial communication data register (SCDR).

SECTION 7TIME PROCESSOR UNIT

The time processor unit (TPU) is an intelligent, semi-autonomous microcontroller designed for timing control. Operating simultaneously with the CPU, the TPU schedules tasks, processes ROM instructions, accesses shared data with the CPU, and performs input and output. **Figure 7-1** is a simplified block diagram of the TPU.

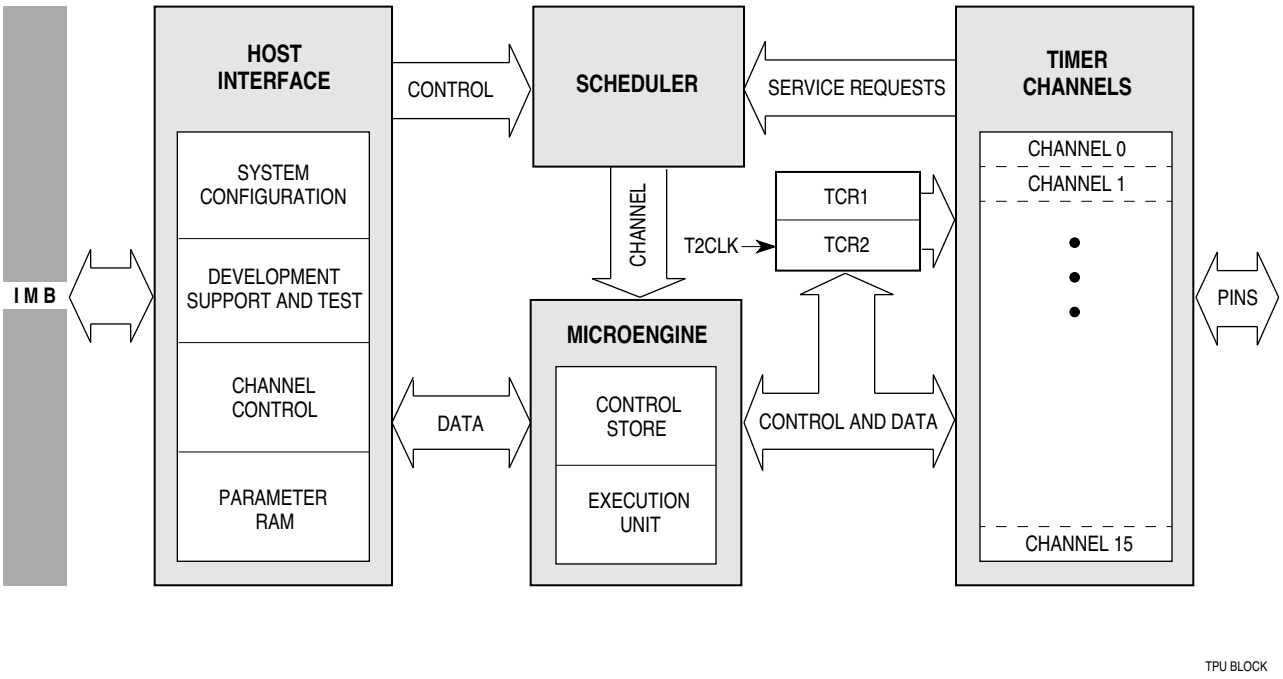


Figure 7-1 TPU Block Diagram

7.1 General

The TPU can be viewed as a special-purpose microcomputer that performs a programmable series of two operations, match and capture. Each occurrence of either operation is called an event. A programmed series of events is called a function. TPU functions replace software functions that would require host CPU interrupt service. The following pre-programmed timing functions are currently available:

- Input capture/input transition counter
- Output compare
- Pulse-width modulation
- Synchronized pulse-width modulation
- Period measurement with additional transition detect
- Period measurement with missing transition detect
- Position-synchronized pulse generator
- Stepper motor
- Period/pulse-width accumulator

7.2 TPU Components

The TPU module consists of two 16-bit time bases, sixteen independent timer channels, a task scheduler, a microengine, and a host interface. In addition, a dual-port parameter RAM is used to pass parameters between the module and the host CPU.

7.2.1 Time Bases

Two 16-bit counters provide reference time bases for all output compare and input capture events. Prescalers for both time bases are controlled by the host CPU via bit fields in the TPU module configuration register (TPUMCR). Timer count registers TCR1 and TCR2 provide access to current counter values. TCR1 and TCR2 can be read or written to by TPU microcode, but are not directly available to the host CPU. The TCR1 clock is derived from the system clock. The TCR2 clock can be derived from the system clock or from an external clock input via the T2CLK pin.

7.2.2 Timer Channels

The TPU has 16 independent channels, each connected to an MCU pin. The channels have identical hardware. Each channel consists of an event register and pin control logic. The event register contains a 16-bit capture register, a 16-bit compare/match register, and a 16-bit greater-than-or-equal-to comparator. The direction of each pin, either output or input, is determined by the TPU microengine. Each channel can either use the same time base for match and capture, or can use one time base for match and the other for capture.

7.2.3 Scheduler

When a service request is received, the scheduler determines which TPU channel is serviced by the microengine. A channel can request service for one of four reasons: for host service, for a link to another channel, for a match event, or for a capture event. The host system assigns each active channel one of three priorities: high, middle, or low. When multiple service requests are received simultaneously, a priority-scheduling mechanism grants service based on channel number and assigned priority.

7.2.4 Microengine

The microengine is composed of a control store and an execution unit. Control-store ROM holds the microcode for each factory-masked time function. When assigned to a channel by the scheduler, the execution unit executes microcode for a function assigned to that channel by the host CPU. Microcode can also be executed from the TPURAM module instead of the control store. The TPURAM module allows emulation and development of custom TPU microcode without the generation of a microcode ROM mask. Refer to **7.3.6 Emulation Support** for more information.

7.2.5 Host Interface

Host interface registers allow communication between the host CPU and the TPU, both before and during execution of a time function. The registers are accessible from the IMB through the TPU bus interface unit. Refer to **7.6 Host Interface Registers** and **APPENDIX D REGISTER SUMMARY** for register bit/field definitions and address

7.2.6 Parameter RAM

Parameter RAM occupies 256 bytes at the top of the system address map. Channel parameters are organized as 128 16-bit words. Although all parameter word locations in RAM can be accessed by all channels, only 100 are normally used: channels 0 to 13 use six parameter words, while channels 14 and 15 each use eight parameter words. The parameter RAM address map in **APPENDIX D REGISTER SUMMARY** shows how parameter words are organized in memory.

The host CPU specifies function parameters by writing the appropriate RAM address. The TPU reads the RAM to determine channel operation. The TPU can also store information to be read by the CPU in RAM. Detailed descriptions of the parameters required by each time function are beyond the scope of this manual. Refer to the *TPU Reference Manual* (TPURM/AD) for more information.

For pre-programmed functions, one of the parameter words associated with each channel contains three channel control fields. These fields perform the following functions:

PSC — Forces the output level of the pin.

PAC — For input capture, PAC specifies the edge transition to be detected. For output comparison, PAC specifies the logic level to be output when a match occurs.

TBS — Specifies channel direction (input or output) and assigns a time base to the input capture and output compare functions of the channel.

7.3 TPU Operation

All TPU functions are related to one of the two 16-bit time bases. Functions are synthesized by combining sequences of match events and capture events. Because the primitives are implemented in hardware, the TPU can determine precisely when a match or capture event occurs, and respond rapidly. An event register for each channel provides for simultaneity of match/capture event occurrences on all channels.

When a match or input capture event requiring service occurs, the affected channel generates a service request to the scheduler. The scheduler determines the priority of the request and assigns the channel to the microengine at the first available time. The microengine performs the function defined by the content of the control store or emulation RAM, using parameters from the parameter RAM.

7.3.1 Event Timing

Match and capture events are handled by independent channel hardware. This provides an event accuracy of one time-base clock period, regardless of the number of channels that are active. An event normally causes a channel to request service. However, before an event can be serviced, any pending previous requests must be serviced. The time needed to respond to and service an event is determined by the number of channels requesting service, the relative priorities of the channels requesting service, and the microcode execution time of the active functions. Worst-case

event service time (latency) determines TPU performance in a given application. Latency can be closely estimated — refer to Freescale *TPU Reference Manual* (TPURM/AD) for more information.

7.3.2 Channel Orthogonality

Most timer systems are limited by the fixed number of functions assigned to each pin. All TPU channels contain identical hardware and are functionally equivalent in operation, so that any channel can be configured to perform any time function. Any function can operate on the calling channel, and, under program control, on another channel determined by the program or by a parameter. The user controls the combination of time functions.

7.3.3 Interchannel Communication

The autonomy of the TPU is enhanced by the ability of a channel to affect the operation of one or more other channels without CPU intervention. Interchannel communication can be accomplished by issuing a link service request to another channel, by controlling another channel directly, or by accessing the parameter RAM of another channel.

7.3.4 Programmable Channel Service Priority

The TPU provides a programmable service priority level to each channel. Three priority levels are available. When more than one channel of a given priority requests service at the same time, arbitration is accomplished according to channel number. To prevent a single high-priority channel from permanently blocking other functions, other service requests of the same priority are performed in channel order after the lowest-numbered, highest-priority channel is serviced.

7.3.5 Coherency

For data to be coherent, all available portions of it must be identical in age, or must be logically related. As an example, consider a 32-bit counter value that is read and written as two 16-bit words. The 32-bit value is read-coherent only if both 16-bit portions are updated at the same time, and write-coherent only if both portions take effect at the same time. Parameter RAM hardware supports coherent access of two adjacent 16-bit parameters. The host CPU must use a long-word operation to guarantee coherency.

7.3.6 Emulation Support

Although factory-programmed time functions can perform a wide variety of control tasks, they may not be ideal for all applications. The TPU provides emulation capability that allows the user to develop new time functions. Emulation mode is entered by setting the EMU bit in the TPUMCR. In emulation mode, an auxiliary bus connection is made between TPURAM and the TPU module, and access to TPURAM via the intermodule bus is disabled. A 9-bit address bus, a 32-bit data bus, and control lines transfer information between the modules. To ensure exact emulation, RAM module access timing remains consistent with access timing of the TPU ROM control store.

To support changing TPU application requirements, Freescale has established a TPU function library. The function library is a collection of TPU functions written for easy assembly in combination with each other or with custom functions. Refer to Freescale Programming Note TPUPN00/D, *Using the TPU Function Library and TPU Emulation Mode* for information about developing custom functions and accessing the TPU function library. Refer to the *TPU Reference Manual* (TPURM/AD) for more information about specific functions.

7.3.7 TPU Interrupts

Each of the TPU channels can generate an interrupt service request. Interrupts for each channel must be enabled by writing to the appropriate control bit in the channel interrupt enable register (CIER). The channel interrupt status register (CISR) contains one interrupt status flag per channel. Time functions set the flags. Setting a flag bit causes the TPU to make an interrupt service request if the corresponding channel interrupt enable bit is set and the interrupt request level is nonzero.

The value of the channel interrupt request level (CIRL) field in TICR determines the priority of all TPU interrupt service requests. CIRL values correspond to MCU interrupt request signals $\overline{\text{IRQ}}$[7:1]. $\overline{\text{IRQ7}}$ is the highest-priority request signal; $\overline{\text{IRQ1}}$ has the lowest priority. Assigning a value of %111 to CIRL causes $\overline{\text{IRQ7}}$ to be asserted when a TPU interrupt request is made; lower field values cause corresponding lower-priority interrupt request signals to be asserted. Assigning CIRL a value of %000 disables all interrupts.

The CPU recognizes only interrupt requests of a priority greater than the value contained in the interrupt priority (IP) mask in the condition code register. When the CPU acknowledges an interrupt request, the priority of the acknowledged interrupt is written to the IP mask and is driven out onto the IMB address lines.

When the IP mask value driven out on the address lines is the same as the CIRL value, the TPU contends for arbitration priority. The IARB field in TPUMCR contains the TPU arbitration number. Each module that can make an interrupt service request must be assigned a unique non-zero IARB value in order to implement an arbitration scheme. Arbitration is performed by means of serial assertion of IARB field bit values. IARB is initialized to $0 during reset.

When the TPU wins arbitration, it must respond to the CPU interrupt acknowledge cycle by placing an interrupt vector number on the data bus. The vector number is used to calculate displacement into the exception vector table. Vectors are formed by concatenating the 4-bit value of the CIBV field in the TPU interrupt configuration register with the 4-bit number of the channel requesting interrupt service. Since the CIBV field has a reset value of %00, it must be assigned a value corresponding to the upper nibble of a block of 16 user-defined vector numbers before TPU interrupts are enabled, or a TPU interrupt service request could cause the CPU to take one of the reserved vectors in the exception vector table.

Refer to **SECTION 4 SYSTEM INTEGRATION MODULE** for further information about interrupts. For more information about the exception vector table refer to **SECTION 5 CENTRAL PROCESSING UNIT**.

7.4 Standard and Enhanced Standard Time Functions

The following paragraphs describe factory-programmed time functions implemented in standard and enhanced standard TPU microcode ROM. A complete description of the functions is beyond the scope of this manual. Refer to the *TPU Reference Manual* (TPURM/AD) for additional information.

7.4.1 Discrete Input/Output (DIO)

When a pin is used as a discrete input, a parameter indicates the current input level and the previous 15 levels of a pin. Bit 15, the most significant bit of the parameter, indicates the most recent state. Bit 14 indicates the next most recent state, and so on. The programmer can choose one of the three following conditions to update the parameter: 1) when a transition occurs, 2) when the CPU makes a request, or 3) when a rate specified in another parameter is matched. When a pin is used as a discrete output, it is set high or low only upon request by the CPU.

7.4.2 Input Capture/Input Transition Counter (ITC)

Any channel of the TPU can capture the value of a specified TCR upon the occurrence of each transition or specified number of transitions, and then generate an interrupt request to notify the CPU. A channel can perform input captures continually, or a channel can detect a single transition or specified number of transitions, then cease channel activity until reinitialization. After each transition or specified number of transitions, the channel can generate a link to a sequential block of up to eight channels. The user specifies a starting channel of the block and the number of channels within the block. The generation of links depends on the mode of operation. In addition, after each transition or specified number of transitions, one byte of the parameter RAM (at an address specified by channel parameter) can be incremented and used as a flag to notify another channel of a transition.

7.4.3 Output Compare (OC)

The output compare function generates a rising edge, falling edge, or a toggle of the previous edge in one of three ways:

1. Immediately upon CPU initiation, thereby generating a pulse with a length equal to a programmable delay time.
2. At a programmable delay time from a user-specified time.
3. Continuously. Upon receiving a link from a channel, OC references, without CPU interaction, a specifiable period and calculates an offset:

$$\text{Offset} = \text{Period} \times \text{Ratio}$$

where Ratio is a parameter supplied by the user.

This algorithm generates a 50% duty-cycle continuous square wave with each high/low time equal to the calculated OFFSET. Due to offset calculation, there is an initial link time before continuous pulse generation begins.

7.4.4 Pulse-Width Modulation (PWM)

The TPU can generate a pulse-width modulation waveform with any duty cycle from zero to 100% (within the resolution and latency capability of the TPU). To define the PWM, the CPU provides one parameter that indicates the period and another parameter that indicates the high time. Updates to one or both of these parameters can direct the waveform change to take effect immediately, or coherently beginning at the next low-to-high transition of the pin.

7.4.5 Synchronized Pulse-Width Modulation (SPWM)

The TPU generates a PWM waveform in which the CPU can change the period and/or high time at any time. When synchronized to a time function on a second channel, the synchronized PWM low-to-high transitions have a time relationship to transitions on the second channel.

7.4.6 Period Measurement with Additional Transition Detect (PMA)

This function and the following function are used primarily in toothed-wheel speed-sensing applications, such as monitoring rotational speed of an engine. The period measurement with additional transition detect function allows for a special-purpose 23-bit period measurement. It can detect the occurrence of an additional transition (caused by an extra tooth on the sensed wheel) indicated by a period measurement that is less than a programmable ratio of the previous period measurement.

Once detected, this condition can be counted and compared to a programmable number of additional transitions detected before TCR2 is reset to $FFFF. Alternatively, a byte at an address specified by a channel parameter can be read and used as a flag. A nonzero value of the flag indicates that TCR2 is to be reset to $FFFF once the next additional transition is detected

7.4.7 Period Measurement with Missing Transition Detect (PMM)

Period measurement with missing transition detect allows a special-purpose 23-bit period measurement. It detects the occurrence of a missing transition (caused by a missing tooth on the sensed wheel), indicated by a period measurement that is greater than a programmable ratio of the previous period measurement. Once detected, this condition can be counted and compared to a programmable number of additional transitions detected before TCR2 is reset to $FFFF. In addition, one byte at an address specified by a channel parameter can be read and used as a flag. A nonzero value of the flag indicates that TCR2 is to be reset to $FFFF once the next missing transition is detected.

7.4.8 Position-Synchronized Pulse Generator (PSP)

Any channel of the TPU can generate an output transition or pulse, which is a projection in time based on a reference period previously calculated on another channel. Both TCRs are used in this algorithm: TCR1 is internally clocked, and TCR2 is clocked by a position indicator in the user's device. An example of a TCR2 clock source is a sensor that detects special teeth on the flywheel of an automobile using PMA or PMM. The teeth are placed at known degrees of engine rotation; hence, TCR2 is a coarse representation of engine degrees, i.e., each count represents some number of degrees.

Up to 15 position-synchronized pulse generator function channels can operate with a single input reference channel executing a PMA or PMM input function. The input channel measures and stores the time period between the flywheel teeth and resets TCR2 when the engine reaches a reference position. The output channel uses the period calculated by the input channel to project output transitions at specific engine degrees. Because the flywheel teeth might be 30 or more degrees apart, a fractional multiplication operation resolves down to the desired degrees. Two modes of operation allow pulse length to be determined either by angular position or by time.

7.4.9 Stepper Motor (SM)

The stepper motor control algorithm provides for linear acceleration and deceleration control of a stepper motor with a programmable number of step rates of up to 14. Any group of channels, up to eight, can be programmed to generate the control logic necessary to drive a stepper motor.

The time period between steps (P) is defined as:

$$P(r) = K1 - K2 \times r$$

where r is the current step rate (1–14), and K1 and K2 are supplied as parameters.

After providing the desired step position in a 16-bit parameter, the CPU issues a step request. Next, the TPU steps the motor to the desired position through an acceleration/deceleration profile defined by parameters. The parameter indicating the desired position can be changed by the CPU while the TPU is stepping the motor. This algorithm changes the control state every time a new step command is received.

A 16-bit parameter initialized by the CPU for each channel defines the output state of the associated pin. The bit pattern written by the CPU defines the method of stepping, such as full stepping or half stepping. With each transition, the 16-bit parameter rotates one bit. The period of each transition is defined by the programmed step rate.

7.4.10 Period/Pulse-Width Accumulator (PPWA)

The period/pulse-width accumulator algorithm accumulates a 16-bit or 24-bit sum of either the period or the pulse width of an input signal over a programmable number of periods or pulses (from 1 to 255). After an accumulation period, the algorithm can generate a link to a sequential block of up to eight channels. The user specifies a starting channel of the block and number of channels within the block. Generation of links depends on the mode of operation. Any channel can be used to measure an accumulated number of periods of an input signal. A maximum of 24 bits can be used for the accumulation parameter. From 1 to 255 period measurements can be made and summed with the previous measurement(s) before the TPU interrupts the CPU, allowing instantaneous or average frequency measurement, and the latest complete accumulation (over the programmed number of periods).

The pulse width (high-time portion) of an input signal can be measured (up to 24 bits) and added to a previous measurement over a programmable number of periods (1 to 255). This provides an instantaneous or average pulse-width measurement capability,

allowing the latest complete accumulation (over the specified number of periods) to always be available in a parameter. By using the output compare function in conjunction with PPWA, an output signal can be generated that is proportional to a specified input signal. The ratio of the input and output frequency is programmable. One or more output signals with different frequencies, yet proportional and synchronized to a single input signal, can be generated on separate channels.

7.4.11 Quadrature Decode (QDEC)

The quadrature decode function uses two channels to decode a pair of out-of-phase signals in order to present the CPU with directional information and a position value. It is particularly suitable for use with slotted encoders employed in motor control. The function derives full resolution from the encoder signals and provides a 16-bit position counter with rollover/under indication via an interrupt.

The counter in parameter RAM is updated when a valid transition is detected on either one of the two inputs. The counter is incremented or decremented depending on the lead/lag relationship of the two signals at the time of servicing the transition. The user can read or write the counter at any time. The counter is free running, overflowing to $0000 or underflowing to $FFFF depending on direction. The QDEC function also provides a time stamp referenced to TCR1 for every valid signal edge and the ability for the host CPU to obtain the latest TCR1 value. This feature allows position interpolation by the host CPU between counts at very slow count rates.

7.5 Motion Control Time Functions

The following paragraphs describe factory-programmed time functions implemented in the motion-control microcode ROM. A complete description of the functions is beyond the scope of this manual. Refer to the *TPU Reference Manual* (TPURM/AD) for additional information.

7.5.1 Table Stepper Motor (TSM)

The TSM function provides for acceleration and deceleration control of a stepper motor with a programmable number of step rates up to 58. TSM uses a table in parameter RAM, rather than an algorithm, to define the stepper motor acceleration profile, allowing the user to fully define the profile. In addition, a slew rate parameter allows fine control of the terminal running speed of the motor independent of the acceleration table. The CPU need only write a desired position, and the TPU accelerates, slews, and decelerates the motor to the required position. Full and half step support is provided for two-phase motors. In addition, a slew rate parameter allows fine control of the terminal running speed of the motor independent of the acceleration table.

7.5.2 New Input Capture/Transition Counter (NITC)

Any channel of the TPU can capture the value of a specified TCR or any specified location in parameter RAM upon the occurrence of each transition or specified number of transitions, and then generate an interrupt request to notify the bus master. The times of the most recent two transitions are maintained in parameter RAM. A channel can perform input captures continually, or a channel can detect a single transition or

specified number of transitions, ceasing channel activity until reinitialization. After each transition or specified number of transitions, the channel can generate a link to other channels.

7.5.3 Queued Output Match (QOM)

QOM can generate single or multiple output match events from a table of offsets in parameter RAM. Loop modes allow complex pulse trains to be generated once, a specified number of times, or continuously. The function can be triggered by a link from another TPU channel. In addition, the reference time for the sequence of matches can be obtained from another channel. QOM can generate pulse-width modulated waveforms, including waveforms with high times of 0% or 100%. QOM also allows a TPU channel to be used as a discrete output pin.

7.5.4 Programmable Time Accumulator (PTA)

PTA accumulates a 32-bit sum of the total high time, low time, or period of an input signal over a programmable number of periods or pulses. The accumulation can start on a rising or falling edge. After the specified number of periods or pulses, the PTA generates an interrupt request and optionally generates links to other channels.

From 1 to 255 period measurements can be made and summed with the previous measurement(s) before the TPU interrupts the CPU, providing instantaneous or average frequency measurement capability, and the latest complete accumulation (over the programmed number of periods).

7.5.5 Multichannel Pulse-Width Modulation (MCPWM)

MCPWM generates pulse-width modulated outputs with full 0% to 100% duty cycle range independent of other TPU activity. This capability requires two TPU channels plus an external gate for one PWM channel. (A simple one-channel PWM capability is supported by the QOM function.)

Multiple PWMs generated by MCPWM have two types of high time alignment: edge aligned and center aligned. Edge aligned mode uses n + 1 TPU channels for n PWMs; center aligned mode uses 2n + 1 channels. Center aligned mode allows a user defined “dead time” to be specified so that two PWMs can be used to drive an H-bridge without destructive current spikes. This feature is important for motor control applications.

7.5.6 Fast Quadrature Decode (FQD)

FQD is a position feedback function for motor control. It decodes the two signals from a slotted encoder to provide the CPU with a 16-bit free running position counter. FQD incorporates a “speed switch” which disables one of the channels at high speed, allowing faster signals to be decoded. A time stamp is provided on every counter update to allow position interpolation and better velocity determination at low speed or when low resolution encoders are used. The third index channel provided by some encoders is handled by the ITC function.

7.5.7 Universal Asynchronous Receiver/Transmitter (UART)

The UART function uses one or two TPU channels to provide asynchronous communications. Data word length is programmable from 1 to 14 bits. The function supports detection or generation of even, odd, and no parity. Baud rate is freely programmable and can be higher than 100 Kbaud. Eight bidirectional UART channels running in excess of 9600 baud can be implemented.

7.5.8 Brushless Motor Commutation (COMM)

This function generates the phase commutation signals for a variety of brushless motors, including three-phase brushless direct current. It derives the commutation state directly from the position decoded in FQD, thus eliminating the need for hall effect sensors.

The state sequence is implemented as a user-configurable state machine, thus providing a flexible approach with other general applications. A CPU offset parameter is provided to allow all the switching angles to be advanced or retarded on the fly by the CPU. This feature is useful for torque maintenance at high speeds.

7.5.9 Frequency Measurement (FQM)

FQM counts the number of input pulses to a TPU channel during a user-defined window period. The function has single shot and continuous modes. No pulses are lost between sample windows in continuous mode. The user selects whether to detect pulses on the rising or falling edge. This function is intended for high speed measurement; measurement of slow pulses with noise rejection can be made with PTA.

7.5.10 Hall Effect Decode (HALLD)

This function decodes the sensor signals from a brushless motor, along with a direction input from the CPU, into a state number. The function supports two- or three-sensor decoding. The decoded state number is written into a COMM channel, which outputs the required commutation drive signals. In addition to brushless motor applications, the function can have more general applications, such as decoding “option” switches.

7.6 Host Interface Registers

The TPU memory map contains three groups of registers:

- System Configuration Registers
- Channel Control and Status Registers
- Development Support and Test Verification Registers

All registers except the channel interrupt status register (CISR) must be read or written by means of word accesses. The address space of the TPU memory map occupies 512 bytes. Unused registers within the 512-byte address space return zeros when read.

7.6.1 System Configuration Registers

The TPU configuration control registers, TPUMCR and TICR, determine the value of the prescaler, perform emulation control, specify whether the external TCR2 pin functions as a clock source or as gate of the DIV8 clock for TCR2, and determine interrupt request level and interrupt vector number assignment. Refer to **APPENDIX D REGISTER SUMMARY** for more information about TPUMCR and TICR.

7.6.1.1 Prescaler Control for TCR1

Timer control register one (TCR1) is clocked from the output of a prescaler. Two fields in the TPUMCR control TCR1. The prescaler's input is the internal TPU system clock divided by either 4 or 32, depending on the value of the PSCK (prescaler clock) bit. The prescaler divides this input by 1, 2, 4, or 8, depending on the value of TCR1P (timer count register 1 prescaler control). Channels using TCR1 have the capability to resolve down to the TPU system clock divided by 4. Refer to **Figure 7-2** and **Table 7-1**.

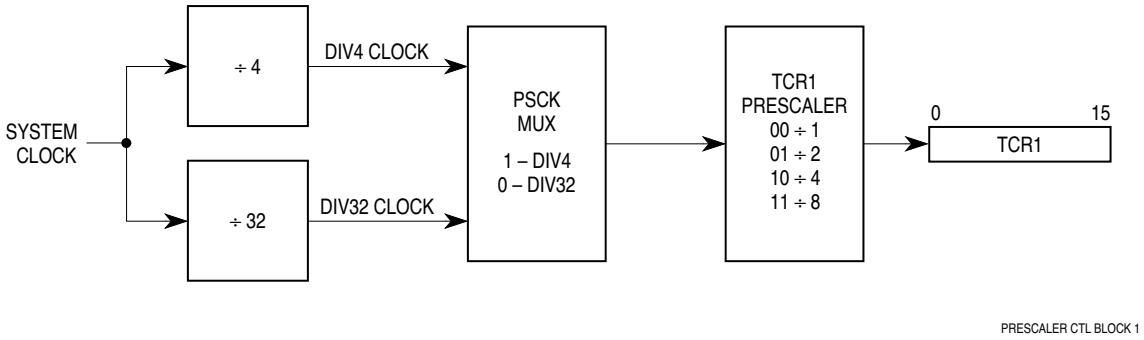


Figure 7-2 TCR1 Prescaler Control

Table 7-1 TCR1 Prescaler Control

TCR1 Prescaler	Divide By	PSCK = 0		PSCK = 1	
		Number of Clocks	Rate at 16 MHz	Number of Clocks	Rate at 16 MHz
00	1	32	2 ms	4	250 ns
01	2	64	4 ms	8	500 ns
10	4	128	8 ms	16	1 ms
11	8	256	16 ms	32	2 ms

7.6.1.2 Prescaler Control for TCR2

Timer control register two (TCR2), like TCR1, is clocked from the output of a prescaler. The T2CG (TCR2 clock/gate control) bit in TPUMCR determines whether the external TCR2 pin functions as an external clock source for TCR2 or as the gate in the use of TCR2 as a gated pulse accumulator. The function of the T2CG bit is shown in **Figure 7-3**.

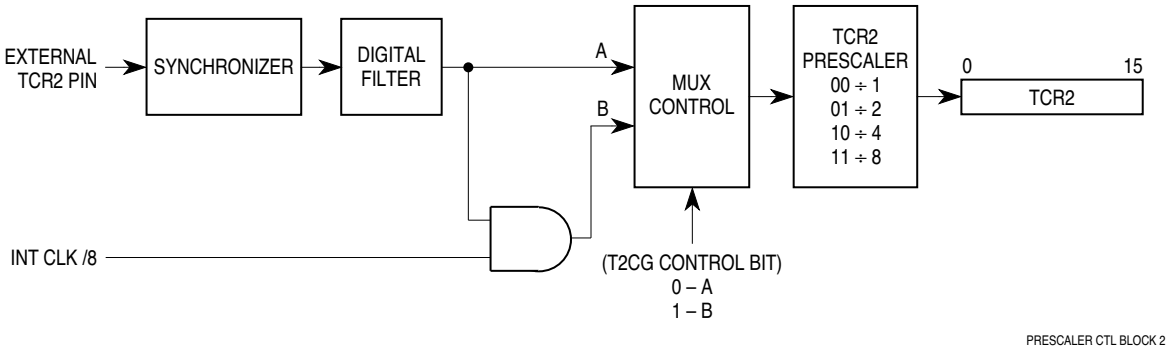


Figure 7-3 TCR2 Prescaler Control

When the T2CG bit is set, the external TCR2 pin functions as a gate of the DIV8 clock (the TPU system clock divided by 8). In this case, when the external TCR2 pin is low, the DIV8 clock is blocked, preventing it from incrementing TCR2. When the external TCR2 pin is high, TCR2 is incremented at the frequency of the DIV8 clock. When T2CG is cleared, an external clock from the TCR2 pin, which has been synchronized and fed through a digital filter, increments TCR2.

The TCR2 field in TPUMCR specifies the value of the prescaler: 1, 2, 4, or 8. Channels using TCR2 have the capability to resolve down to the TPU system clock divided by 8. **Table 7-2** is a summary of prescaler output.

Table 7-2 TCR2 Prescaler Control

TCR2 Prescaler	Divide By	Internal Clock Divided By	External Clock Divided By
00	1	8	1
01	2	16	2
10	4	32	4
11	8	64	8

7.6.1.3 Emulation Control

Asserting the EMU bit in the TPUMCR places the TPU in emulation mode. In emulation mode, the TPU executes microinstructions from TPURAM exclusively. Access to the TPURAM module through the IMB by a host is blocked, and the TPURAM module is dedicated for use by the TPU. After reset, EMU can be written only once.

7.6.1.4 Low-Power Stop Control

If the STOP bit in the TPUMCR is set, the TPU shuts down its internal clocks, shutting down the internal microengine. TCR1 and TCR2 cease to increment and retain the last value before the stop condition was entered. The TPU asserts the stop flag (STF) in the TPUMCR to indicate that it has stopped.

7.6.2 Channel Control Registers

The channel control and status registers enable the TPU to control channel interrupts, assign time functions to be executed on a specified channel, or select the mode of operation or the type of host service request for the time function specified. Refer to **Table 7-3**.

7.6.2.1 Channel Interrupt Enable and Status Registers

The channel interrupt enable register (CIER) allows the CPU to enable or disable the ability of individual TPU channels to request interrupt service. Setting the appropriate bit in the register enables a channel to make an interrupt service request; clearing a bit disables the interrupt.

The channel interrupt status register (CISR) contains one interrupt status flag per channel. Time functions specify via microcode when an interrupt flag is set. Setting a flag causes the TPU to make an interrupt service request if the corresponding CIER bit is set and the CIRL field has a nonzero value. To clear a status flag, read CISR, then write a zero to the appropriate bit. CISR is the only TPU register that can be accessed on a byte basis.

7.6.2.2 Channel Function Select Registers

Encoded 4-bit fields within the channel function select registers specify one of 16 time functions to be executed on the corresponding channel. Encodings for predefined functions in the TPU ROM are found in **Table 7-3**.

7.6.2.3 Host Sequence Registers

The host sequence field selects the mode of operation for the time function selected on a given channel. The meaning of the host sequence bits depends on the time function specified. Refer to **Table 7-3**, which is a summary of the host sequence and host service request bits for each time function.

7.6.2.4 Host Service Registers

The host service request field selects the type of host service request for the time function selected on a given channel. The meaning of the host service request bits is determined by time function microcode.

A host service request field cleared to %00 signals the host that service is completed by the microengine on that channel. The host can request service on a channel by writing the corresponding host service request field to one of three nonzero states. It is a good practice to monitor the host service request register and wait until the TPU clears the service request before changing any parameters or issuing a new service request to the channel.

7.6.2.5 Channel Priority Registers

The channel priority registers (CPR1, CPR2) assign one of three priority levels to a channel or disable the channel. **Table 7-3** indicates the number of time slots guaranteed for each channel priority encoding.

Table 7-3 Channel Priority Encodings

CHX[1:0]	Service	Guaranteed Time Slots
00	Disabled	—
01	Low	1 out of 7
10	Middle	2 out of 7
11	High	4 out of 7

7.6.3 Development Support and Test Registers

These registers are used for custom microcode development or for factory test. Describing the use of the registers is beyond the scope of this manual. Register descriptions are provided in **APPENDIX D REGISTER SUMMARY**. Refer to the *TPU Reference Manual* (TPURM/AD) for more information.

SECTION 8 STANDBY RAM WITH TPU EMULATION

The standby RAM module with TPU emulation capability (TPURAM) consists of a control register block and a 2-Kbyte array of fast (two bus cycle) static RAM, which is especially useful for system stacks and variable storage. The TPURAM responds to both program and data space accesses. The TPURAM can be used to emulate TPU microcode ROM.

8.1 General

The TPURAM can be mapped to any 2-Kbyte boundary in the address map, but must not overlap the module control registers. Refer to **8.3 TPURAM Array Address Mapping** for more information. Data can be read or written in bytes, words or long words. The TPURAM is powered by V_{DD} in normal operation. During power-down, TPURAM contents can be maintained by power from the V_{STBY} input. Power switching between sources is automatic.

8.2 TPURAM Register Block

There are three TPURAM control registers: the TPURAM module configuration register (TRAMMCR), the TPURAM test register (TRAMTST), and the TPURAM base address and status register (TRAMBAR). To protect these registers from accidental modification, they are always mapped to supervisor data space.

The TPURAM control register block begins at address $7FFB00 or $FFFB00, depending on the value of the module mapping (MM) bit in the SIM configuration register (SIMCR). **SECTION 4 SYSTEM INTEGRATION MODULE** contains more information about how the state of MM affects the system.

There is a 64-byte minimum control register block size for the TPURAM module. Unimplemented register addresses are read as zeros, and writes have no effect. Refer to **APPENDIX D REGISTER SUMMARY** for the register block address map and register bit/field definitions.

8.3 TPURAM Array Address Mapping

Base address and status register TRAMBAR specifies the TPURAM array base address in the MCU memory map. TRAMBAR[15:3] specify the 13 MSBs of the base address. The TPU bus interface unit compares these bits to address lines ADDR[23:11]. If the two match, then the low order address lines and the SIZ[1:0] signals are used to access the RAM location in the array. The TPURAM can be mapped to any 2-Kbyte boundary in the address map, but must not overlap the module control registers. Overlap makes the registers inaccessible.

The RAM disable (RAMDS) bit, the LSB of the TRAMBAR, indicates whether the TPURAM array is active (RAMDS = 0) or disabled (RAMDS = 1). The array is disabled coming out of reset and remains disabled if the base address field is programmed with

an address that overlaps the address of the module control register block. Writing a valid base address to TRAMBAR[15:3] clears RAMDS and enables the array.

TRAMBAR can be written only once after a master reset. This prevents runaway software from accidentally re-mapping the array. Because the locking mechanism is activated by the first write after a master reset, the base address field should be written in a single word operation. Writing only one-half of the register prevents the other half from being written. Note that in test mode the locking mechanism for TRAMBAR can be disabled by the RTBA bit in the TRAMTST register.

8.4 TPURAM Privilege Level

The RASP field in TRAMMCR specifies whether access to the TPURAM module can be made from the supervisor privilege level only or from either the user or supervisor privilege level. If supervisor-only access is specified, an access from the user privilege level is ignored by the TPURAM control logic and can be decoded externally. Refer to **SECTION 4 SYSTEM INTEGRATION MODULE** and **SECTION 5 CENTRAL PROCESSING UNIT** for more information concerning privilege levels.

8.5 Normal Operation

In normal operation, TPURAM is accessed via the IMB by a bus master and is powered by V_{DD}. The array can be accessed by byte, word, or long word. A byte or aligned word access takes one bus cycle (two system clock cycles). A long word access requires two bus cycles. Refer to **SECTION 4 SYSTEM INTEGRATION MODULE** for more information concerning access times.

During normal operation, the TPU does not access the array and has no effect on the operation of the TPURAM module.

8.6 Standby Operation

Standby mode maintains the RAM array when the MCU main power supply is turned off. Low-power mode allows the central processing unit to control MCU power consumption.

Relative voltage levels of the V_{DD} and V_{STBY} pins determine whether the TPURAM is in standby mode. TPURAM circuitry switches to the standby power source when specified limits are exceeded. If specified standby supply voltage levels are maintained during the transition, there is no loss of memory when switching occurs. The RAM array cannot be accessed while the TPURAM module is powered from V_{STBY}. If standby operation is not desired, connect the V_{STBY} pin to the V_{SS} pin.

I_{SB} exceeds specified maximum standby current during the time V_{DD} makes the transition from normal operating level to the level specified for standby operation. This occurs within the voltage range $V_{SB} - 0.5\ V \geq V_{DD} \geq V_{SS} + 0.5\ V$. Typically, I_{SB} peaks when V_{DD}» $V_{SB} - 1.5\ V$, and averages 1.0 mA over the transition period.

Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for standby switching and power consumption specifications.

To prevent standby supply voltage from going below the specified minimum, a filter capacitor must be attached between the V_{STBY} and V_{SS} pins. To calculate filter capac-

itance, V_{DD} supply ramp time, available standby voltage, and available standby current must be known. Assuming that the rate of change is constant as V_{DD} changes from 0.0 V to 5.5 V (nominal V_{SS} to nominal V_{DD}) and that V_{SB} also drops during this period, capacitance is calculated using the following expression:

$$C = \frac{It}{V}$$

Where:

C = Desired capacitance
I = I_{SB} differential (Transient I_{SB} – Available supply current)
t = time of maximum I_{SB} (Typically in the range V_{SB} – 1.5 V ± 0.5 V)
V = V_{SB} differential (Available supply voltage – Specified minimum V_{SB})

8.7 Low-Power Stop Operation

Setting the STOP bit in the TRAMMCR switches the TPURAM module to low-power mode. In low-power mode, the array retains its contents, but cannot be read or written by the CPU. STOP can be written only when the processor is operating at the supervisor privilege level. STOP is set during reset. Stop mode is exited by clearing STOP.

The TPURAM module will switch to standby mode while it is in low-power mode, provided the operating constraints discussed above are met.

8.8 Reset

Reset places the TPURAM in low-power mode, enables supervisor-level access only, clears the base address, and disables the array. These actions make it possible to write a new base address into the base address register.

When a synchronous reset occurs while a byte or word TPURAM access is in progress, the access is completed. If reset occurs during the first word access of a long-word operation, only the first word access is completed. If reset occurs during the second word access of a long-word operation, the entire access is completed. Data being read from or written to the TPURAM may be corrupted by asynchronous reset. Refer to **SECTION 4 SYSTEM INTEGRATION MODULE** for more information concerning resets.

8.9 TPU Microcode Emulation

The TPURAM array can emulate the microcode ROM in the TPU module. This provides a means of developing custom TPU code. The TPU selects TPU emulation mode.

The TPU is connected to the TPURAM via a dedicated bus. While the TPURAM array is in TPU emulation mode, the access timing of the TPURAM module matches the timing of the TPU microinstruction ROM to ensure accurate emulation. Normal accesses through the IMB are inhibited and the control registers have no effect, allowing external RAM to emulate the TPURAM at the same addresses. Refer to **SECTION 7 TIME PROCESSOR UNIT** and to the *TPU Reference Manual* (TPURM/AD) for more information.

APPENDIX A ELECTRICAL CHARACTERISTICS

This appendix contains electrical specification tables and reference timing diagrams.

Table A-1 Maximum Ratings

Num	Rating	Symbol	Value	Unit
1	Supply Voltage[1, 2, 7]	V_{DD}	–0.3 to +6.5	V
2	Input Voltage[1, 2, 3, 5, 7]	V_{in}	–0.3 to +6.5	V
3	Instantaneous Maximum Current Single pin limit (applies to all pins)[1, 5, 6, 7]	I_D	25	mA
4	Operating Maximum Current Digital Input Disruptive Current [4,5,6,7,8] $V_{NEGCLMAP} \cong -0.3$ V $V_{POSCLAMP} \cong V_{DD} + 0.3$	I_{ID}	–500 to 500	µA
5	Operating Temperature Range MC68332 No Suffix MC68332 “C” Suffix MC68332 “V” Suffix MC68332 “M” Suffix	T_A	T_L to T_H 0 to 70 –40 to 85 –40 to 105 –40 to 125	°C
6	Storage Temperature Range	T_{stg}	–55 to 150	°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. All pins except $\overline{\text{TSTME}}$/TSC.
4. All functional non-supply pins are internally clamped to V_{SS}. All functional pins except EXTAL and XFC are internally clamped to V_{DD}.
5. Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values for positive and negative clamp voltages, then use the larger of the two values.
6. Power supply must maintain regulation within operating V_{DD} range during instantaneous and operating maximum current conditions.
7. This parameter is periodically sampled rather than 100% tested.
8. Total input current for all digital input-only and all digital input/output pins must not exceed 10 mA. Exceeding this limit can cause disruption of normal operation.

Table A-2 Typical Ratings, 16.78 MHz Operation

Num	Rating	Symbol	Value	Unit
1	Supply Voltage	V_{DD}	5.0	V
2	Operating Temperature	T_A	25	°C
3	V_{DD} Supply Current RUN LPSTOP, VCO off LPSTOP, External clock, maxi f_{sys}	I_{DD}	 75 125 3	 mA µA mA
4	Clock Synthesizer Operating Voltage	V_{DDSYN}	5.0	V
5	V_{DDSYN} Supply Current VCO on, maximum f_{sys} External Clock, maximum f_{sys} LPSTOP, VCO off V_{DD} powered down	I_{DDSYN}	 1.0 4.0 100 50	 mA mA µA µA
6	RAM Standby Voltage	V_{SB}	3.0	V
7	RAM Standby Current Normal RAM operation Standby operation	I_{SB}	 7.0 40	 µA µA
8	Power Dissipation	P_D	455	mW

Table A-2a. Typical Ratings, 20.97 MHz Operation

Num	Rating	Symbol	Value	Unit
1	Supply Voltage	V_{DD}	5.0	V
2	Operating Temperature	T_A	25	°C
3	V_{DD} Supply Current RUN LPSTOP, VCO off LPSTOP, External clock, maxi f_{sys}	I_{DD}	 113 125 3.75	 mA µA mA
4	Clock Synthesizer Operating Voltage	V_{DDSYN}	5.0	V
5	V_{DDSYN} Supply Current VCO on, maximum f_{sys} External Clock, maximum f_{sys} LPSTOP, VCO off V_{DD} powered down	I_{DDSYN}	 1.0 5.0 100 50	 mA mA µA µA
6	RAM Standby Voltage	V_{SB}	3.0	V
7	RAM Standby Current Normal RAM operation Standby operation	I_{SB}	 7.0 40	 µA µA
8	Power Dissipation	P_D	570	mW

Table A-3 Thermal Characteristics

Num	Rating	Symbol	Value	Unit
1	Thermal Resistance Plastic 132-Pin Surface Mount Plastic 144-Pin Surface Mount Thin Plastic 144-Pin Surface Mount	Θ_{JA}	 38 46 49	°C/W

Notes:

The average chip-junction temperature (T_J) in C can be obtained from:

$$T_J = T_A + (P_D \times \Theta_{JA}) \quad (1)$$

where

T_A = Ambient Temperature, °C

Θ_{JA} = Package Thermal Resistance, Junction-to-Ambient, °C/W

P_D = $P_{INT} + P_{I/O}$

P_{INT} = $I_{DD} \times V_{DD}$, Watts — Chip Internal Power

$P_{I/O}$ = Power Dissipation on Input and Output Pins — User Determined

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:

$$P_D = K \div (T_J + 273°C) \quad (2)$$

Solving equations 1 and 2 for K gives:

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

where K is a constant pertaining to the particular part. K can be determined from equation (3) by measuring P_D (at equilibrium) for a known T_A. Using this value of K, the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.

Table A-4 16.78 MHz Clock Control Timing

(V_{DD} and V_{DDSYN} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, 32.768 kHz reference)

Num	Characteristic	Symbol	Min	Max	Unit
1	PLL Reference Frequency Range	f_{ref}	25	50	kHz
2	System Frequency[1] On-Chip PLL System Frequency External Clock Operation	f_{sys}	dc 0.131 dc	16.78 16.78 16.78	MHz
3	PLL Lock Time[2,3,4,5]	t_{lpll}	—	20	ms
4	VCO Frequency[6]	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 Stability[2,3,4,7] Short term (5 µs interval) Long term (500 µs interval)	C_{stab}	 –0.5 –0.05	 0.5 0.05	%

Table A-4a. 20.97 MHz Clock Control Timing

(V_{DD} and V_{DDSYN} = 5.0 Vdc ±5%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, 32.768 kHz reference)

Num	Characteristic	Symbol	Min	Max	Unit
1	PLL Reference Frequency Range	f_{ref}	25	50	kHz
2	System Frequency[1] On-Chip PLL System Frequency External Clock Operation	f_{sys}	dc 0.131 dc	20.97 20.97 20.97	MHz
3	PLL Lock Time[2,3,4,5]	t_{lpll}	—	20	ms
4	VCO Frequency[6]	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 Stability[2,3,4,7] Short term (5 µs interval) Long term (500 µs interval)	C_{stab}	 –0.5 –0.05	 0.5 0.05	%

Notes For Tables 4 And 4a:

1. All internal registers retain data at 0 Hz.
2. This parameter is periodically sampled rather than 100% tested.
3. 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 Ω to guarantee this specification. Filter network geometry can vary depending upon operating environment (See **4.3 System Clock**).
4. Proper layout procedures must be followed to achieve specifications.
5. 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 $\overline{\text{RESET}}$ 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.
6. 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}.
7. Stability 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 C_{stab} percentage for a given interval. When clock stability is a critical constraint on control system operation, this parameter should be measured during functional testing of the final system.

Table A-5 16.78 MHz DC Characteristics

(V_{DD} and V_{DDSYN} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ 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[1]	V_{HYS}	0.5	—	V
4	Input Leakage Current[2] V_{in} = V_{DD} or V_{SS} Input-only pins	I_{in}	–2.5	2.5	µA
5	High Impedance (Off-State) Leakage Current[2] V_{in} = V_{DD} or V_{SS}All input/output and output pins	I_{OZ}	 –2.5	 2.5	µA
6	CMOS Output High Voltage[2, 3] I_{OH} = –10.0 µAGroup 1, 2, 4 input/output and all output pins	V_{OH}	V_{DD} – 0.2	—	V
7	CMOS Output Low Voltage[2] I_{OL} = 10.0 µAGroup 1, 2, 4 input/output and all output pins	V_{OL}	—	0.2	V
8	Output High Voltage[2, 3] I_{OH} = –0.8 mAGroup 1, 2, 4 input/output and all output pins	V_{OH}	V_{DD} – 0.8	—	V
9	Output Low Voltage[2] I_{OL} = 1.6 mAGroup 1 I/O Pins, CLKOUT, FREEZE/QUOT, $\overline{\text{IPIPE}}$ I_{OL} = 5.3 mAGroup 2 and Group 4 I/O Pins, $\overline{\text{CSBOOT}}$, BG/$\overline{\text{CS}}$ I_{OL} = 12 mAGroup 3	V_{OL}	 — — —	 0.4 0.4 0.4	V
10	Three State Control Input High Voltage	V_{IHTSC}	1.6 (V_{DD})	9.1	V
11	Data Bus Mode Select Pull-up Current[5] V_{in} = V_{IL}DATA[15:0] V_{in} = V_{IH}DATA[15:0]	I_{MSP}	 — –15	 –120 —	µA
12	V_{DD} Supply Current[6] RUN [4] RUN, TPU emulation mode LPSTOP, 32.768 kHz crystal, VCO Off (STSIM = 0) LPSTOP (External clock input frequency = maximum f_{sys})	 I_{DD} I_{DD} S_{IDD} S_{IDD}	 — — — —	 124 134 350 5	 mA mA µA mA
13	Clock Synthesizer Operating Voltage	V_{DDSYN}	4.5	5.5	V
14	V_{DDSYN} Supply Current[6] 32.768 kHz crystal, VCO on, maximum f_{sys} External Clock, maximum f_{sys} LPSTOP, 32.768 kHz crystal, VCO off (STSIM = 0) 32.768 kHz crystal, V_{DD} powered down	 I_{DDSYN} I_{DDSYN} S_{IDDSYN} I_{DDSYN}	 — — — —	 1 5 150 100	 mA mA µA µA
15	RAM Standby Voltage[7] Specified V_{DD} applied V_{DD} = V_{SS}	V_{SB}	 0.0 3.0	 5.5 5.5	V
16	RAM Standby Current[6,7,10] Normal RAM operation$V_{DD} > V_{SB}$ – 0.5 V Transient conditionV_{SB} – 0.5 V ≥ V_{DD} ≥ V_{SS} + 0.5 V Standby operation$V_{DD} < V_{SS}$ + 0.5 V	I_{SB}	 — — —	 10 3 60	 µA mA µA
17	Power Dissipation[8]	P_D	—	690	mW
18	Input Capacitance[2, 9]All input-only pins All input/output pins	C_{in}	— —	10 20	pF
19	Load Capacitance[2] Group 1 I/O Pins and CLKOUT, FREEZE/QUOT, $\overline{\text{IPIPE}}$ Group 2 I/O Pins and $\overline{\text{CSBOOT}}$, BG/$\overline{\text{CS}}$ Group 3 I/O pins Group 4 I/O pins	C_L	 — — —	 90 100 130 200	pF

Table A-5a. 20.97 MHz DC Characteristics

(V_{DD} and V_{DDSYN} = 5.0 Vdc ± 5%, V_{SS} = 0 Vdc, $T_A = T_L$ 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[1]	V_{HYS}	0.5	—	V
4	Input Leakage Current[2] V_{in} = V_{DD} or V_{SS} Input-only pins	I_{in}	–2.5	2.5	µA
5	High Impedance (Off-State) Leakage Current[2] V_{in} = V_{DD} or V_{SS}All input/output and output pins	I_{OZ}	 –2.5	 2.5	µA
6	CMOS Output High Voltage[2, 3] I_{OH} = –10.0 µAGroup 1, 2, 4 input/output and all output pins	V_{OH}	V_{DD} – 0.2	—	V
7	CMOS Output Low Voltage[2] I_{OL} = 10.0 µAGroup 1, 2, 4 input/output and all output pins	V_{OL}	—	0.2	V
8	Output High Voltage[2, 3] I_{OH} = –0.8 mAGroup 1, 2, 4 input/output and all output pins	V_{OH}	V_{DD} – 0.8	—	V
9	Output Low Voltage[2] I_{OL} = 1.6 mAGroup 1 I/O Pins, CLKOUT, FREEZE/QUOT, $\overline{IPIPE}$ I_{OL} = 5.3 mAGroup 2 and Group 4 I/O Pins, $\overline{CSBOOT}$, BG/$\overline{CS}$ I_{OL} = 12 mAGroup 3	V_{OL}	 — — —	 0.4 0.4 0.4	V
10	Three State Control Input High Voltage	V_{IHTSC}	1.6 (V_{DD})	9.1	V
11	Data Bus Mode Select Pull-up Current[5] V_{in} = V_{IL}DATA[15:0] V_{in} = V_{IH}DATA[15:0]	I_{MSP}	 — –15	 –120 —	µA
12	V_{DD} Supply Current[6] RUN [4] RUN, TPU emulation mode LPSTOP, 32.768 kHz crystal, VCO Off (STSIM = 0) LPSTOP (External clock input frequency = maximum f_{sys})	 I_{DD} I_{DD} S_{IDD} S_{IDD}	 — — — —	 140 150 350 5	 mA mA µA mA
13	Clock Synthesizer Operating Voltage	V_{DDSYN}	4.75	5.25	V
14	V_{DDSYN} Supply Current[6] 32.768 kHz crystal, VCO on, maximum f_{sys} External Clock, maximum f_{sys} LPSTOP, 32.768 kHz crystal, VCO off (STSIM = 0) 32.768 kHz crystal, V_{DD} powered down	 I_{DDSYN} I_{DDSYN} S_{IDDSYN} I_{DDSYN}	 — — — —	 2 6 150 100	 mA mA µA µA
15	RAM Standby Voltage[7] Specified V_{DD} applied V_{DD} = V_{SS}	V_{SB}	 0.0 3.0	 5.25 5.25	V
16	RAM Standby Current[6,7,10] Normal RAM operationV_{DD} > V_{SB} – 0.5 V Transient conditionV_{SB} – 0.5 V ≥ V_{DD} ≥ V_{SS} + 0.5 V Standby operationV_{DD} < V_{SS} + 0.5 V	I_{SB}	 — —	 10 3 50	 µA mA µA
17	Power Dissipation[8]	P_D	—	766	mW
18	Input Capacitance[2, 9]All input-only pins All input/output pins	C_{in}	— —	10 20	pF
19	Load Capacitance[2] Group 1 I/O Pins and CLKOUT, FREEZE/QUOT, $\overline{IPIPE}$ Group 2 I/O Pins and $\overline{CSBOOT}$, BG/$\overline{CS}$ Group 3 I/O pins Group 4 I/O pins	C_L	 — — —	 90 100 130 200	pF

Notes for Tables A–5 and A–5a:

1. Applies to:
   Port E [7:4] — SIZ[1:0], AS, DS
   Port F [7:0] — IRQ[7:1], MODCLK
   Port QS [7:0] — TXD, PCS[3:1],ÊPCS0/SS, SCK, MOSI, MISO
   TPUCH[15:0], T2CLK
   BKPT/DSCLK, IFETCH, RESET, RXD, TSTME/TSC
   EXTAL (when PLL enabled)
2. Input-Only Pins: EXTAL, TSTME/TSC, BKPT, T2CLK, RXD
   Output-Only Pins: CSBOOT, BG/CS, CLKOUT, FREEZE/QUOT, IPIPE
   Input/Output Pins:
   Group 1: DATA[15:0], IFETCH, TPUCH[15:0]
   Group 2: Port C [6:0] — ADDR[22:19]/CS[9:6], FC[2:0]/CS[5:3]
   Port E [7:0] — SIZ[1:0], AS, DS, AVEC, RMC, DSACK[1:0]
   Port F [&:0] — IRQ[7:1], MODCLK
   Port QS [7:3] — TXD, PCS[3:1], PCS0/SS
   ADDR23/CS10/ECLK, ADDR[18:0], R/W, BERR, BR/CS0, BGACK/CS2
   Group 3: HALT, RESET
   Group 4: MISO, MOSI, SCK
3. Does not apply to HALT and RESET because they are open drain pins. Does not apply to Port QS [7:0] (TXD, PCS[3:1],ÊPCS0/SS, SCK, MOSI, MISO) in wired-OR mode.
4. Current measured with system clock frequency of 16.78 MHz, all modules active.
5. Use of an active pulldown device is recommended.
6. Total operating current is the sum of the appropriate I_{DD}, I_{DDSYN}, and I_{SB} values. I_{DD} values include supply currents for device modules powered by V_{DDE} and V_{DDI} pins.
7. The RAM module will not switch into standby mode as long as V_{SB} does not exceed V_{DD} by more than 0.5 Volt. The RAM array cannot be accessed while the module is in standby mode.
8. Power dissipation measured at specified system clock frequency, all modules active. Power dissipation can be calculated using the expression:
   $P_D = \text{Maximum } V_{DD} (I_{DD} + I_{DDSYN} + I_{SB})$
   I_{DD} includes supply currents for all device modules powered by V_{DDE} and V_{DDI} pins.
9. This parameter is periodically sampled rather than 100% tested.
10. When V_{DD} is transitioning during power-up or power down sequence, and V_{SB} is applied, current flows between the V_{STBY} and V_{DD} pins, which causes standby current to increase toward the maximum transient condition specification. System noise on the V_{DD} and V_{STBY} pins can contribute to this condition.

Table A-6 16.78 MHz AC Timing

(V_{DD} and V_{DDSYN} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)

Num	Characteristic	Symbol	Min	Max	Unit
F1	Frequency of Operation (32.768 kHz crystal)[2]	f	0.13	16.78	MHz
1	Clock Period	t_{cyc}	59.6	—	ns
1A	ECLK Period	t_{Ecyc}	476	—	ns
1B	External Clock Input Period[3]	t_{Xcyc}	59.6	—	ns
2, 3	Clock Pulse Width	t_{CW}	24	—	ns
2A, 3A	ECLK Pulse Width	t_{ECW}	236	—	ns
2B, 3B	External Clock Input High/Low Time[3]	t_{XCHL}	29.8	—	ns
4, 5	Clock Rise and Fall Time	t_{Crf}	—	5	ns
4A, 5A	Rise and Fall Time — All Outputs except CLKOUT	t_{rf}	—	8	ns
4B, 5B	External Clock Rise and Fall Time[4]	t_{XCrf}	—	5	ns
6	Clock High to Address, FC, SIZE, $\overline{RMC}$ Valid	t_{CHAV}	0	29	ns
7	Clock High to Address, Data, FC, SIZE, $\overline{RMC}$ High Impedance	t_{CHAZx}	0	59	ns
8	Clock High to Address, 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)[5]	t_{STSA}	–15	15	ns
9C	Clock Low to $\overline{IFETCH}$, $\overline{IPIPE}$ Asserted	t_{CLIA}	2	22	ns
11	Address, FC, SIZE, $\overline{RMC}$ Valid to $\overline{AS}$, $\overline{CS}$ Asserted	t_{AVSA}	15	—	ns
12	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated	t_{CLSN}	2	29	ns
12A	Clock Low to $\overline{IFETCH}$, $\overline{IPIPE}$ Negated	t_{CLIN}	2	22	ns
13	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to Address, FC, SIZE Invalid (Address Hold)	t_{SNAI}	15	—	ns
14	$\overline{AS}$, $\overline{CS}$ Width Asserted	t_{SWA}	100	—	ns
14A	$\overline{DS}$, $\overline{CS}$ Width Asserted (Write)	t_{SWAW}	45	—	ns
14B	$\overline{AS}$, $\overline{CS}$ Width Asserted (Fast Write Cycle)	t_{SWDW}	40	—	ns
15	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Width Negated[6]	t_{SN}	40	—	ns
16	Clock High to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ High Impedance	t_{CHSZ}	—	59	ns
17	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to R/$\overline{W}$ Negated	t_{SNRN}	15	—	ns
18	Clock High to R/$\overline{W}$ High	t_{CHRH}	0	29	ns
20	Clock High to R/$\overline{W}$ Low	t_{CHRL}	0	29	ns
21	R/$\overline{W}$ Asserted to $\overline{AS}$, $\overline{CS}$ Asserted	t_{RAAA}	15	—	ns
22	R/$\overline{W}$ Low to $\overline{DS}$, $\overline{CS}$ Asserted (Write)	t_{RASA}	70	—	ns
23	Clock High to Data Out Valid	t_{CHDO}	—	29	ns
24	Data Out Valid to Negating Edge of $\overline{AS}$, $\overline{CS}$	t_{DVASN}	15	—	ns
25	$\overline{DS}$, $\overline{CS}$ Negated to Data Out Invalid (Data Out Hold)	t_{SNDOI}	15	—	ns
26	Data Out Valid to $\overline{DS}$, $\overline{CS}$ Asserted (Write)	t_{DVSA}	15	—	ns
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}	20	—	ns
28	$\overline{AS}$, $\overline{DS}$ Negated to $\overline{DSACK}$[1:0], $\overline{BERR}$, $\overline{HALT}$, $\overline{AVEC}$ Negated	t_{SNDN}	0	80	ns
29	$\overline{DS}$, $\overline{CS}$ Negated to Data In Invalid (Data In Hold)[7]	t_{SNDI}	0	—	ns
29A	$\overline{DS}$, $\overline{CS}$ Negated to Data In High Impedance[7, 8]	t_{SHDI}	—	55	ns
30	CLKOUT Low to Data In Invalid (Fast Cycle Hold)[7]	t_{CLDI}	15	—	ns

Table A-6 16.78 MHz AC Timing (Continued)

(V_{DD} and V_{DDSYN} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)

Num	Characteristic	Symbol	Min	Max	Unit
30A	CLKOUT Low to Data In High Impedance[7]	t_{CLDH}	—	90	ns
31	$\overline{DSACK}$[1:0] Asserted to Data In Valid[9]	t_{DADI}	—	50	ns
33	Clock Low to $\overline{BG}$ Asserted/Negated	t_{CLBAN}	—	29	ns
35	$\overline{BR}$ Asserted to $\overline{BG}$ Asserted ($\overline{RMC}$ Not Asserted)[10]	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}	150	—	ns
46A	R/$\overline{W}$ Width Asserted (Fast Write or Read Cycle)	t_{RWAS}	90	—	ns
47A	Asynchronous Input Setup Time $\overline{BR}$, $\overline{BGACK}$, $\overline{DSACK}$[1:0], $\overline{BERR}$, $\overline{AVEC}$, $\overline{HALT}$	t_{AIST}	5	—	ns
47B	Asynchronous Input Hold Time	t_{AIHT}	15	—	ns
48	$\overline{DSACK}$[1:0] Asserted to $\overline{BERR}$, $\overline{HALT}$ Asserted[11]	t_{DABA}	—	30	ns
53	Data Out Hold from Clock High	t_{DOCH}	0	—	ns
54	Clock High to Data Out High Impedance	t_{CHDH}	—	28	ns
55	R/$\overline{W}$ Asserted to Data Bus Impedance Change	t_{RADC}	40	—	ns
56	$\overline{RESET}$ 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)	t_{SCLDD}	0	29	ns
71	Data Setup Time to Clock Low (Show)	t_{SCLDS}	15	—	ns
72	Data Hold from Clock Low (Show)	t_{SCLDH}	10	—	ns
73	$\overline{BKPT}$ Input Setup Time	t_{BKST}	15	—	ns
74	$\overline{BKPT}$ Input Hold Time	t_{BKHT}	10	—	ns
75	Mode Select Setup Time	t_{MSS}	20	—	t_{cyc}
76	Mode Select Hold Time	t_{MSH}	0	—	ns
77	$\overline{RESET}$ Assertion Time[12]	t_{RSTA}	4	—	t_{cyc}
78	$\overline{RESET}$ Rise Time[13]	t_{RSTR}	—	10	t_{cyc}

Table A-6a. 20.97 MHz AC Timing

(V_{DD} and V_{DDSYN} = 5.0 Vdc ± 5%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)

Num	Characteristic	Symbol	Min	Max	Unit
F1	Frequency of Operation (32.768 kHz crystal)[2]	f	0.13	20.97	MHz
1	Clock Period	t_{cyc}	47.7	—	ns
1A	ECLK Period	t_{Ecyc}	381	—	ns
1B	External Clock Input Period[3]	t_{Xcyc}	47.7	—	ns
2, 3	Clock Pulse Width	t_{CW}	18.8	—	ns
2A, 3A	ECLK Pulse Width	t_{ECW}	183	—	ns
2B, 3B	External Clock Input High/Low Time[3]	t_{XCHL}	23.8	—	ns
4, 5	Clock Rise and Fall Time	t_{Crf}	—	5	ns
4A, 5A	Rise and Fall Time — All Outputs except CLKOUT	t_{rf}	—	8	ns
4B, 5B	External Clock Rise and Fall Time[4]	t_{XCrf}	—	5	ns
6	Clock High to Address, FC, SIZE, $\overline{RMC}$ Valid	t_{CHAV}	0	23	ns
7	Clock High to Address, Data, FC, SIZE, $\overline{RMC}$ High Impedance	t_{CHAZx}	0	47	ns
8	Clock High to Address, FC, SIZE, $\overline{RMC}$ Invalid	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Asserted	t_{CLSA}	0	23	ns
9A	$\overline{AS}$ to $\overline{DS}$ or $\overline{CS}$ Asserted (Read)[5]	t_{STSA}	–10	10	ns
9C	Clock Low to $\overline{IFETCH}$, $\overline{IPIPE}$ Asserted	t_{CLIA}	2	22	ns
11	Address, FC, SIZE, $\overline{RMC}$ Valid to $\overline{AS}$, $\overline{CS}$ Asserted	t_{AVSA}	10	—	ns
12	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated	t_{CLSN}	2	23	ns
12A	Clock Low to $\overline{IFETCH}$, $\overline{IPIPE}$ Negated	t_{CLIN}	2	22	ns
13	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to Address, FC, SIZE Invalid (Address Hold)	t_{SNAI}	10	—	ns
14	$\overline{AS}$, $\overline{CS}$ Width Asserted	t_{SWA}	80	—	ns
14A	$\overline{DS}$, $\overline{CS}$ Width Asserted (Write)	t_{SWAW}	36	—	ns
14B	$\overline{AS}$, $\overline{CS}$ Width Asserted (Fast Write Cycle)	t_{SWDW}	32	—	ns
15	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Width Negated[6]	t_{SN}	32	—	ns
16	Clock High to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ High Impedance	t_{CHSZ}	—	47	ns
17	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to R/$\overline{W}$ Negated	t_{SNRN}	10	—	ns
18	Clock High to R/$\overline{W}$ High	t_{CHRH}	0	23	ns
20	Clock High to R/$\overline{W}$ Low	t_{CHRL}	0	23	ns
21	R/$\overline{W}$ Asserted to $\overline{AS}$, $\overline{CS}$ Asserted	t_{RAAA}	10	—	ns
22	R/$\overline{W}$ Low to $\overline{DS}$, $\overline{CS}$ Asserted (Write)	t_{RASA}	54	—	ns
23	Clock High to Data Out Valid	t_{CHDO}	—	23	ns
24	Data Out Valid to Negating Edge of $\overline{AS}$, $\overline{CS}$	t_{DVASN}	10	—	ns
25	$\overline{DS}$, $\overline{CS}$ Negated to Data Out Invalid (Data Out Hold)	t_{SNDOI}	10	—	ns
26	Data Out Valid to $\overline{DS}$, $\overline{CS}$ Asserted (Write)	t_{DVSA}	10	—	ns
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}	15	—	ns
28	$\overline{AS}$, $\overline{DS}$ Negated to $\overline{DSACK}$[1:0], $\overline{BERR}$, $\overline{HALT}$, $\overline{AVEC}$ Negated	t_{SNDN}	0	60	ns
29	$\overline{DS}$, $\overline{CS}$ Negated to Data In Invalid (Data In Hold)[7]	t_{SNDI}	0	—	ns
29A	$\overline{DS}$, $\overline{CS}$ Negated to Data In High Impedance[7, 8]	t_{SHDI}	—	48	ns
30	CLKOUT Low to Data In Invalid (Fast Cycle Hold)[7]	t_{CLDI}	10	—	ns

Table A-6a. 20.97 MHz AC Timing (Continued)

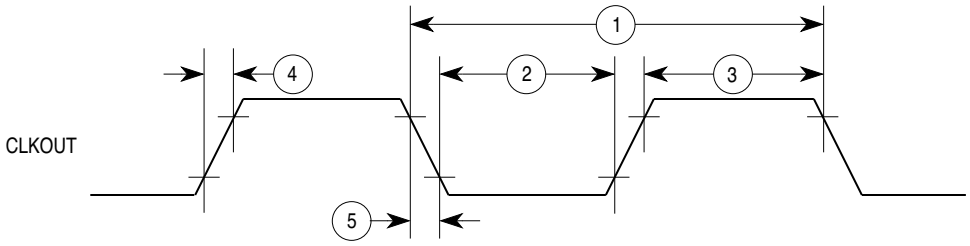
(V_{DD} and V_{DDSYN} = 5.0 Vdc ± 5%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)

Num	Characteristic	Symbol	Min	Max	Unit
30A	CLKOUT Low to Data In High Impedance[7]	t_{CLDH}	—	72	ns
31	$\overline{DSACK}$[1:0] Asserted to Data In Valid[9]	t_{DADI}	—	46	ns
33	Clock Low to $\overline{BG}$ Asserted/Negated	t_{CLBAN}	—	23	ns
35	$\overline{BR}$ Asserted to $\overline{BG}$ Asserted ($\overline{RMC}$ Not Asserted)[10]	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}	115	—	ns
46A	R/$\overline{W}$ Width Asserted (Fast Write or Read Cycle)	t_{RWAS}	70	—	ns
47A	Asynchronous Input Setup Time $\overline{BR}$, $\overline{BGACK}$, $\overline{DSACK}$[1:0], $\overline{BERR}$, $\overline{AVEC}$, $\overline{HALT}$	t_{AIST}	5	—	ns
47B	Asynchronous Input Hold Time	t_{AIHT}	12	—	ns
48	$\overline{DSACK}$[1:0] Asserted to $\overline{BERR}$, $\overline{HALT}$ Asserted[11]	t_{DABA}	—	30	ns
53	Data Out Hold from Clock High	t_{DOCH}	0	—	ns
54	Clock High to Data Out High Impedance	t_{CHDH}	—	23	ns
55	R/$\overline{W}$ Asserted to Data Bus Impedance Change	t_{RADC}	32	—	ns
56	$\overline{RESET}$ 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)	t_{SCLDD}	0	23	ns
71	Data Setup Time to Clock Low (Show)	t_{SCLDS}	10	—	ns
72	Data Hold from Clock Low (Show)	t_{SCLDH}	10	—	ns
73	$\overline{BKPT}$ Input Setup Time	t_{BKST}	10	—	ns
74	$\overline{BKPT}$ Input Hold Time	t_{BKHT}	10	—	ns
75	Mode Select Setup Time	t_{MSS}	20	—	t_{cyc}
76	Mode Select Hold Time	t_{MSH}	0	—	ns
77	$\overline{RESET}$ Assertion Time[12]	t_{RSTA}	4	—	t_{cyc}
78	$\overline{RESET}$ Rise Time[13,14]	t_{RSTR}	—	10	t_{cyc}

Notes for Tables A–6 and A–6a:

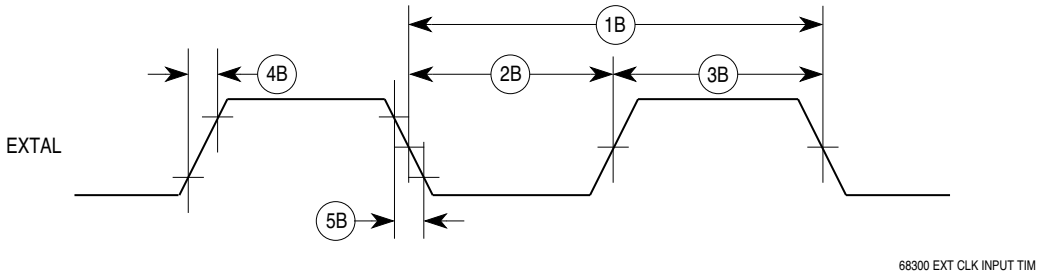
1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
2. Minimum system clock frequency is four times the crystal frequency, subject to specified limits.
3. When an external clock is used, minimum high and low times are based on a 50% duty cycle. The minimum allowable t_{Xcyc} period is reduced when the duty cycle of the external clock signal varies. The relationship between external clock input duty cycle and minimum t_{Xcyc} is expressed:

   Minimum t_{Xcyc} period = minimum t_{XCHL} / (50% – external clock input duty cycle tolerance).
4. Parameters for an external clock signal applied while the internal PLL is disabled (MODCLK pin held low during reset). Does not pertain to an external VCO reference applied while the PLL is enabled (MODCLK pin held high during reset). When the PLL is enabled, the clock synthesizer detects successive transitions of the reference signal. If transitions occur within the correct clock period, rise/fall times and duty cycle are not critical.
5. 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.
6. 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.
7. 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.
8. Maximum value is equal to (t_{cyc} / 2) + 25 ns.
9. 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.
10. To ensure coherency during every operand transfer, $\overline{BG}$ will not be asserted in response to $\overline{BR}$ until after all cycles of the current operand transfer are complete and $\overline{RMC}$ is negated.
11. In the absence of $\overline{DSACK[1:0]}$, $\overline{BERR}$ is an asynchronous input using the asynchronous setup time (specification 47A).
12. After external $\overline{RESET}$ negation is detected, a short transition period (approximately 2 t_{cyc}) elapses, then the SIM drives $\overline{RESET}$ low for 512 t_{cyc}.
13. External assertion of the $\overline{RESET}$ input can overlap internally-generated resets. To insure that an external reset is recognized in all cases, $\overline{RESET}$ must be asserted for at least 590 CLKOUT cycles.
14. External logic must pull $\overline{RESET}$ high during this period in order for normal MCU operation to begin.
15. Address access time = (2.5 + WS) t_{cyc} – t_{CHAV} – t_{DICL}
    Chip select access time = (2 + WS) t_{cyc} – t_{CLSA} – t_{DICL}
    Where: WS = number of wait states. When fast termination is used (2 clock bus) WS = –1.



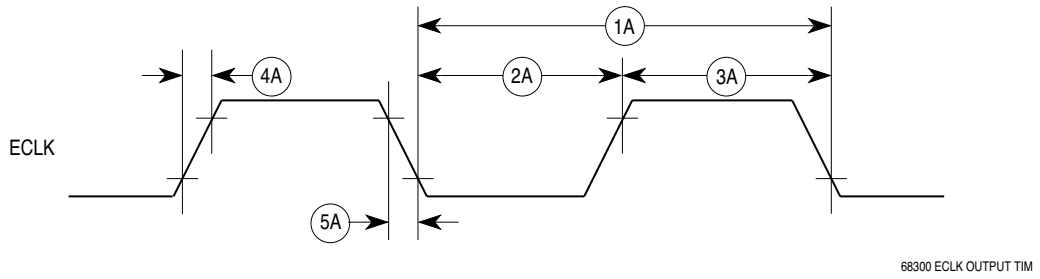
NOTE: Timing shown with respect to 20% and 70% V_{DD}.

Figure A-1 CLKOUT Output Timing Diagram



NOTE: Timing shown with respect to 20% and 70% V_{DD}. Pulse width shown with respect to 50% V_{DD}.

Figure A-2 External Clock Input Timing Diagram



NOTE: Timing shown with respect to 20% and 70% V_{DD}.

Figure A-3 ECLK Output Timing Diagram

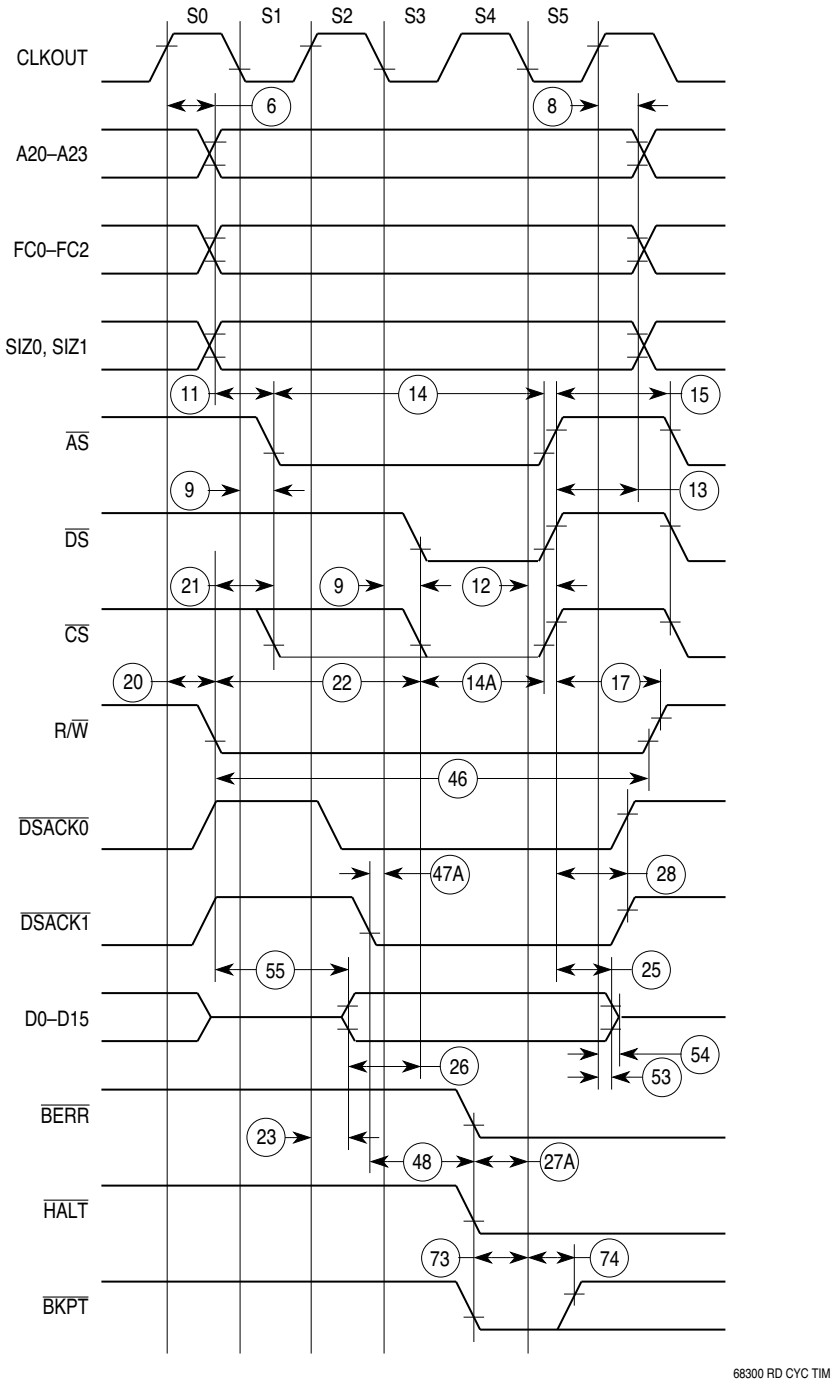


Figure A-4 Read Cycle Timing Diagram

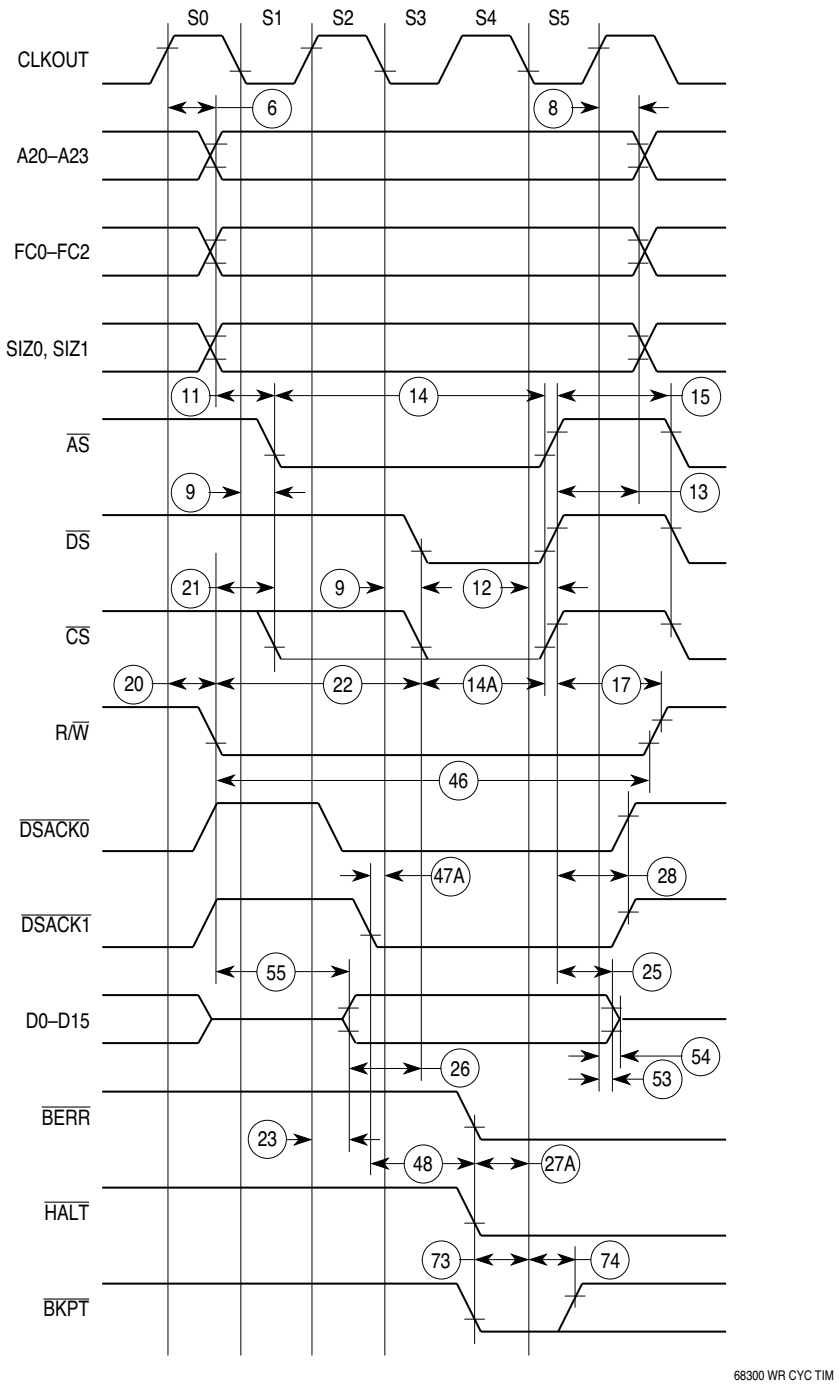


Figure A-5 Write Cycle Timing Diagram

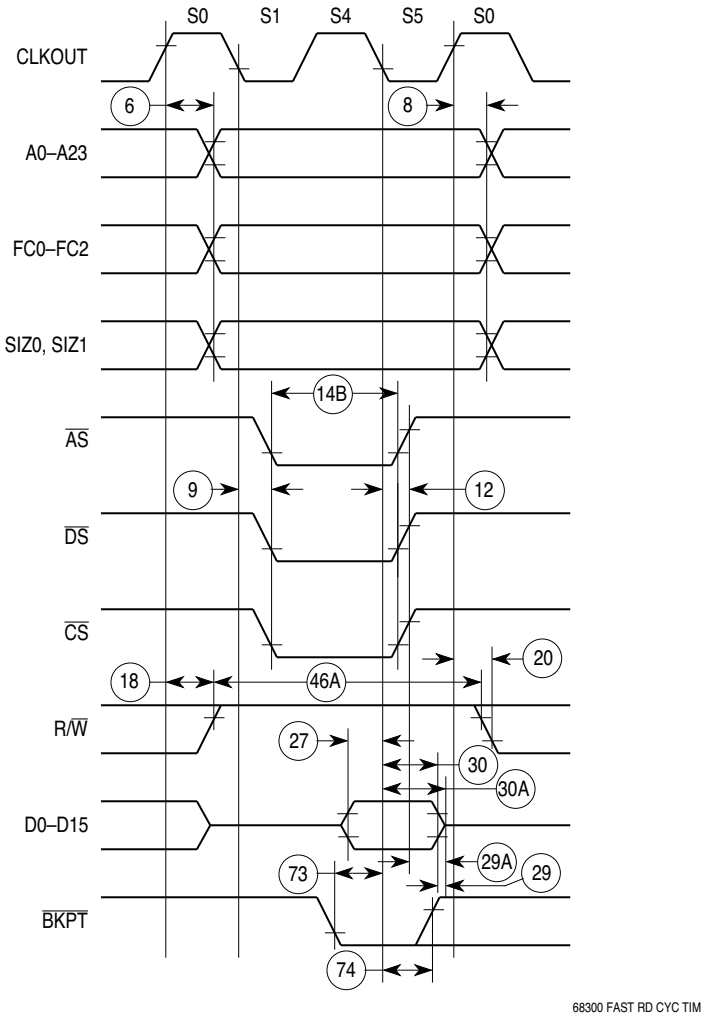


Figure A-6 Fast Termination Read Cycle Timing Diagram

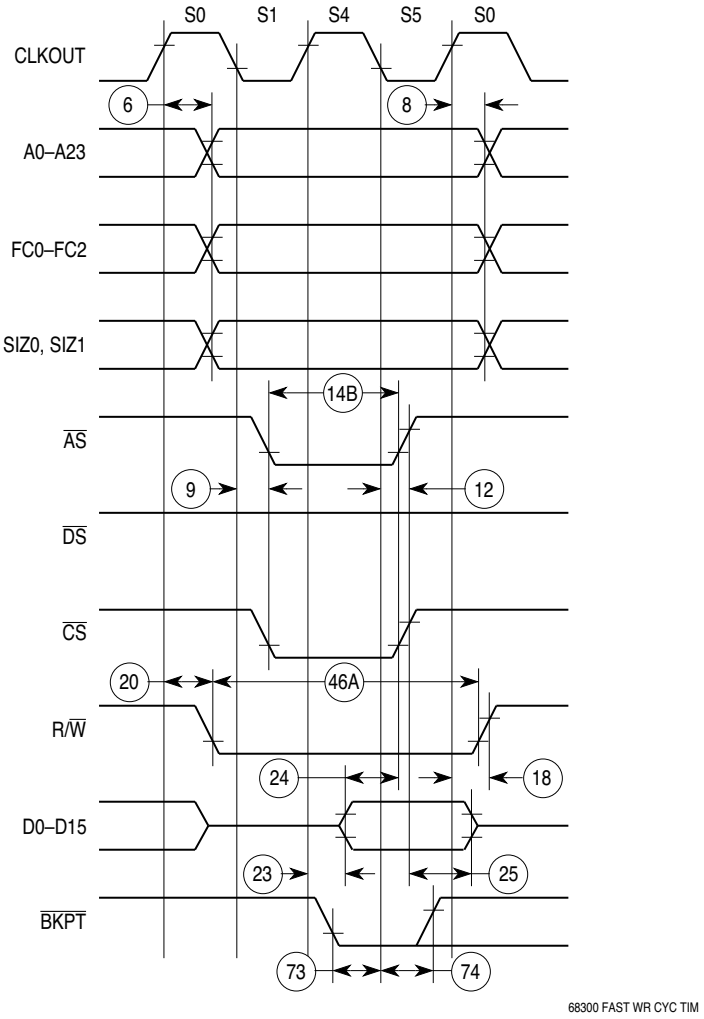


Figure A-7 Fast Termination Write Cycle Timing Diagram

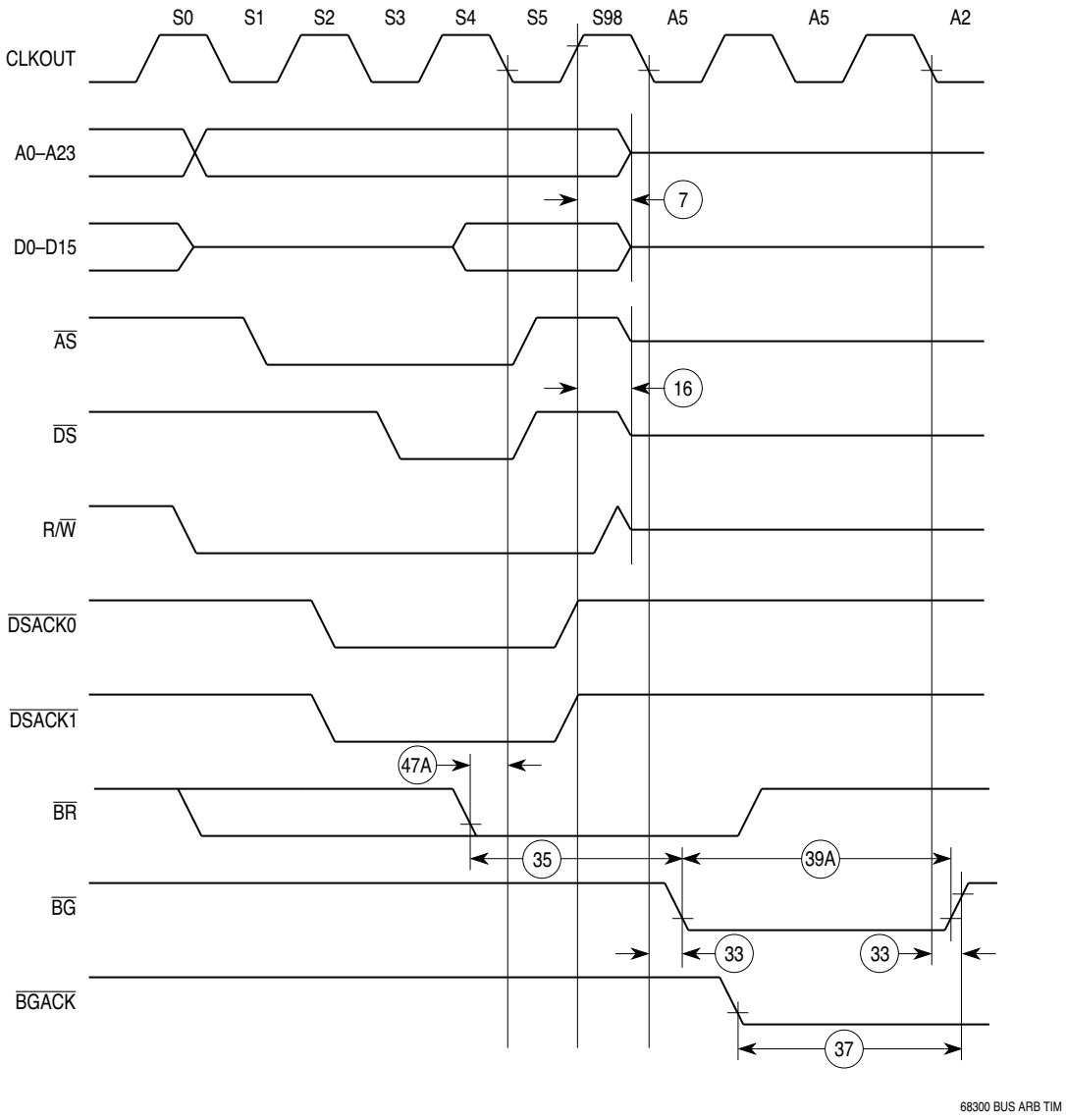


Figure A-8 Bus Arbitration Timing Diagram —Active Bus Case

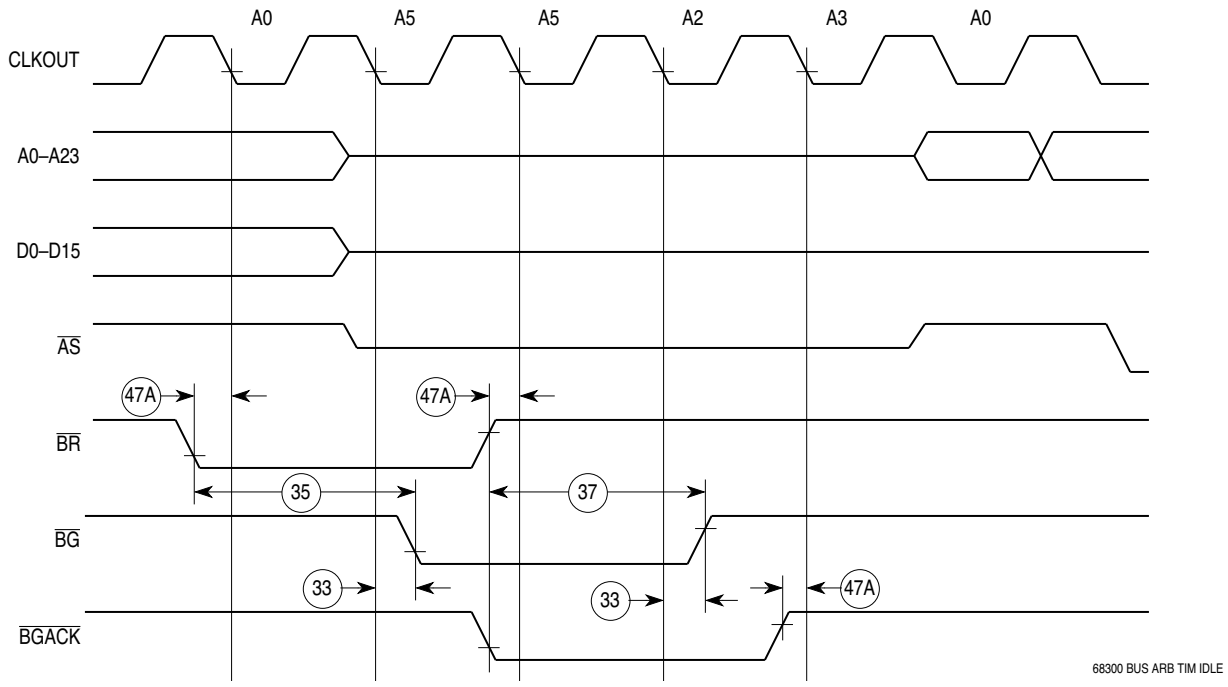
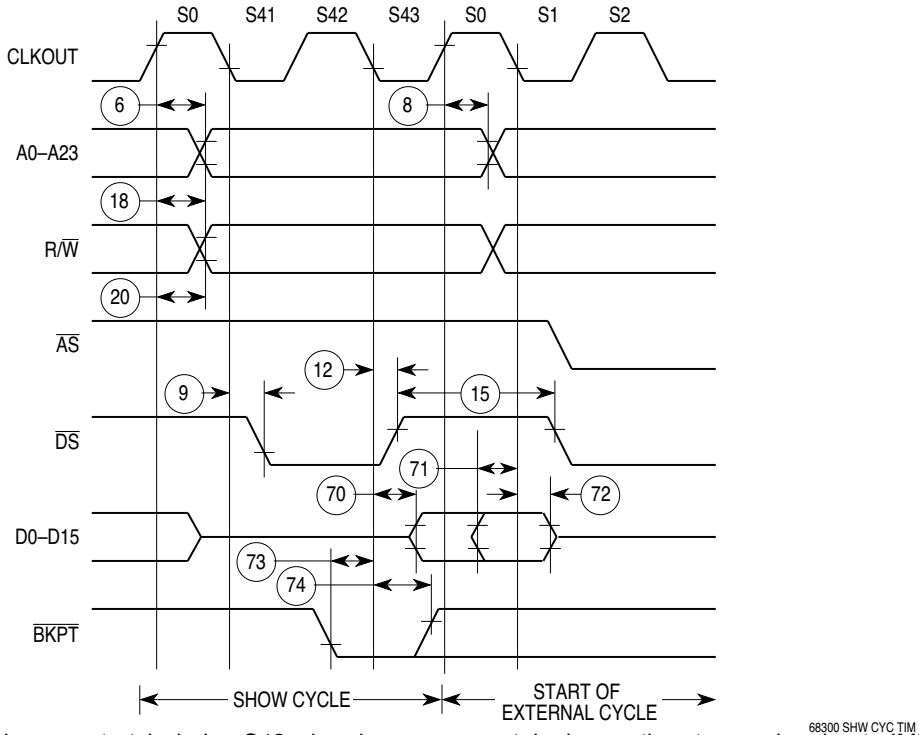
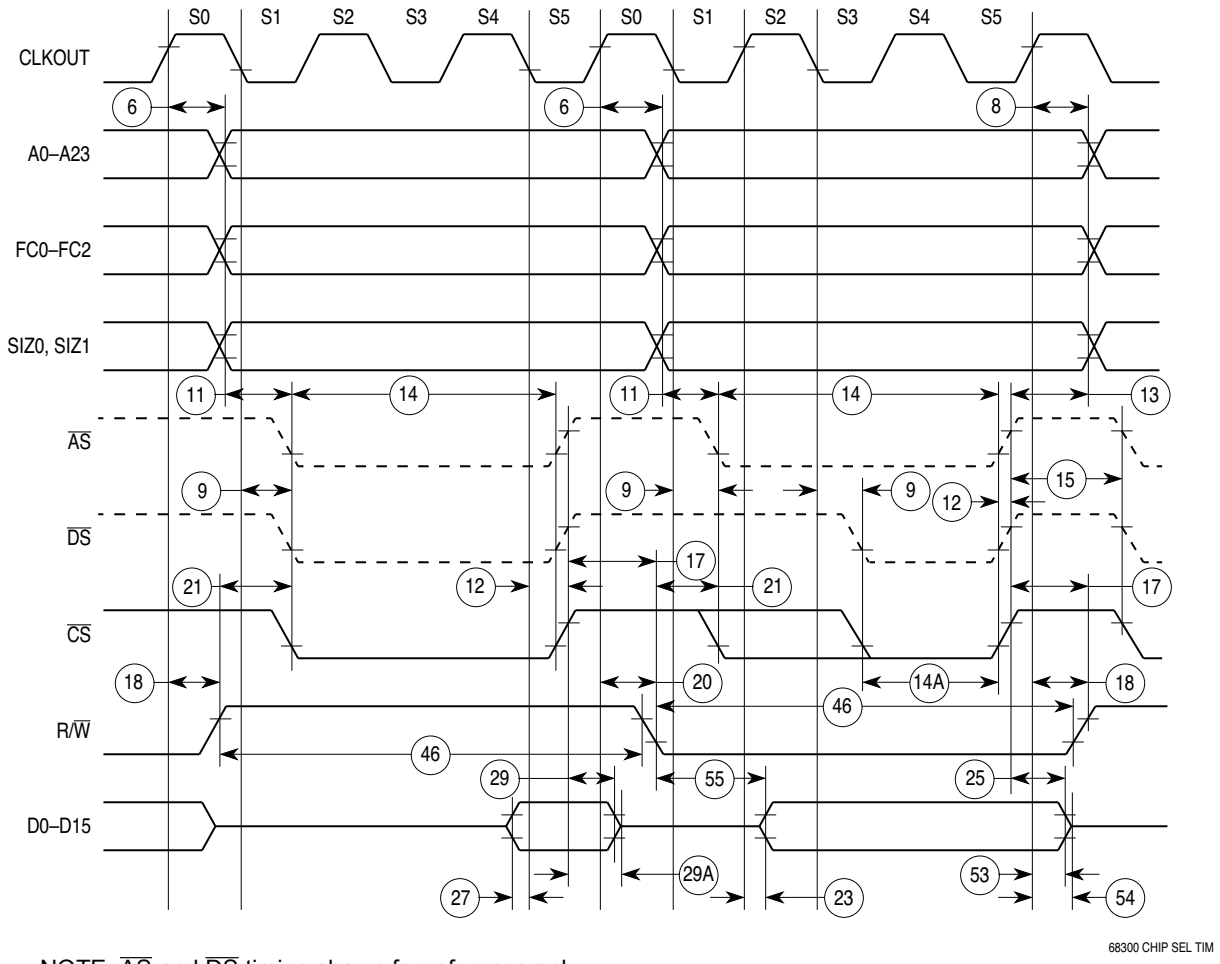


Figure A-9 Bus Arbitration Timing Diagram — Idle Bus Case



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

Figure A-10 Show Cycle Timing Diagram



NOTE: $\overline{AS}$ and $\overline{DS}$ timing shown for reference only.

Figure A-11 Chip Select Timing Diagram

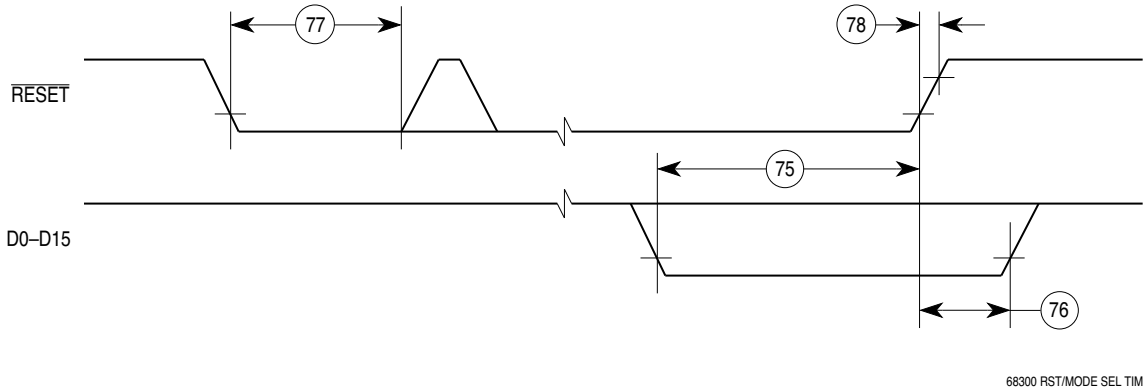


Figure A-12 Reset and Mode Select Timing Diagram

Table A-7 Background Debugging Mode Timing

(V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = T_L to T_H)

Num	Characteristic	Symbol	Min	Max	Unit
B0	DSI Input Setup Time	t_{DSISU}	15	—	ns
B1	DSI Input Hold Time	t_{DSIH}	10	—	ns
B2	DSCLK Setup Time	t_{DSCSU}	15	—	ns
B3	DSCLK Hold Time	t_{DSCH}	10	—	ns
B4	DSO Delay Time	t_{DSOD}	—	25	ns
B5	DSCLK Cycle Time	t_{DSCCYC}	2	—	t_{cyc}
B6	CLKOUT High to FREEZE Asserted/Negated	t_{FRZAN}	—	50	ns
B7	CLKOUT High to $\overline{\text{IFETCH}}$ High Impedance	t_{IFZ}	—	50	ns
B8	CLKOUT High to $\overline{\text{IFETCH}}$ Valid	t_{IF}	—	50	ns
B9	DSCLK Low Time	t_{DSCLO}	1	—	t_{cyc}
B10	FREEZE Asserted to $\overline{\text{IFETCH}}$ Valid	t_{FRZIF}	TBD	—	t_{cyc}

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

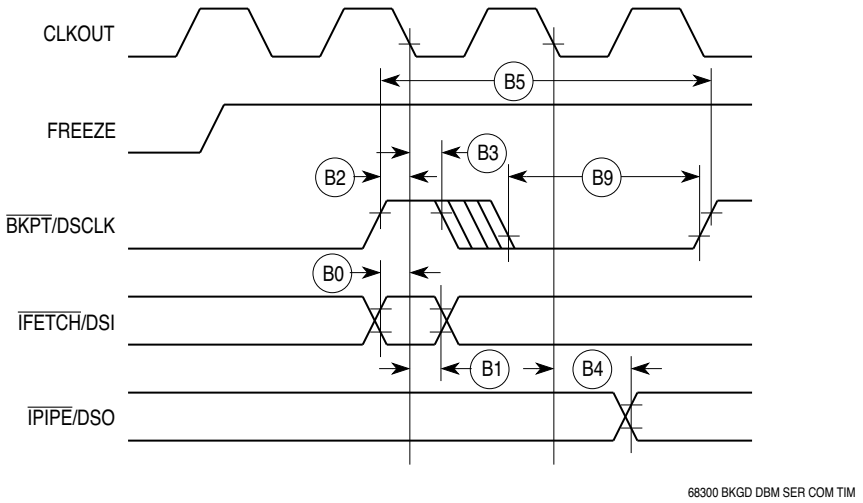


Figure A-13 Background Debugging Mode Timing Diagram — Serial Communication

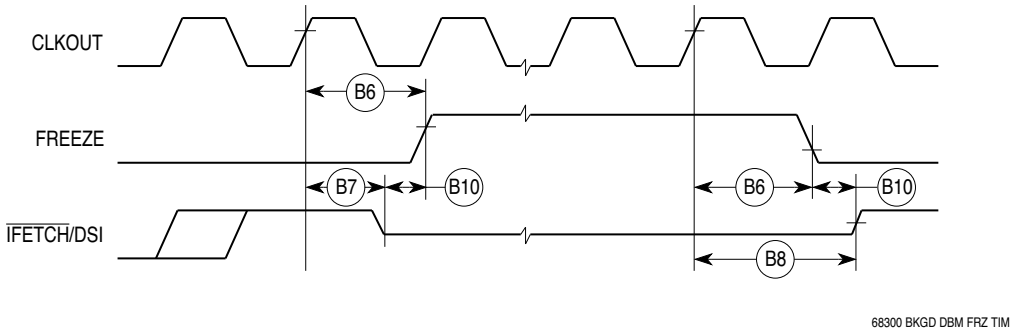


Figure A-14 Background Debugging Mode Timing Diagram — Freeze Assertion

Table A-8 16.78 MHz ECLK Bus Timing

(V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = T_L to T_H)

Num	Characteristic	Symbol	Min	Max	Unit
E1	ECLK Low to Address Valid[2]	t_{EAD}	—	60	ns
E2	ECLK Low to Address Hold	t_{EAH}	15	—	ns
E3	ECLK Low to $\overline{CS}$ Valid ($\overline{CS}$ delay)	t_{ECSD}	—	150	ns
E4	ECLK Low to $\overline{CS}$ Hold	t_{ECSH}	15	—	ns
E5	$\overline{CS}$ Negated Width	t_{ECSN}	30	—	ns
E6	Read Data Setup Time	t_{EDSR}	30	—	ns
E7	Read Data Hold Time	t_{EDHR}	5	—	ns
E8	ECLK Low to Data High Impedance	t_{EDHZ}	—	60	ns
E9	$\overline{CS}$ Negated to Data Hold (Read)	t_{ECDH}	0	—	ns
E10	$\overline{CS}$ Negated to Data High Impedance	t_{ECDZ}	—	1	t_{cyc}
E11	ECLK Low to Data Valid (Write)	t_{EDDW}	—	2	t_{cyc}
E12	ECLK Low to Data Hold (Write)	t_{EDHW}	15	—	ns
E13	Address Access Time (Read)[3]	t_{EACC}	386	—	ns
E14	Chip Select Access Time (Read)[4]	t_{EACS}	296	—	ns
E15	Address Setup Time	t_{EAS}	1/2	—	t_{cyc}

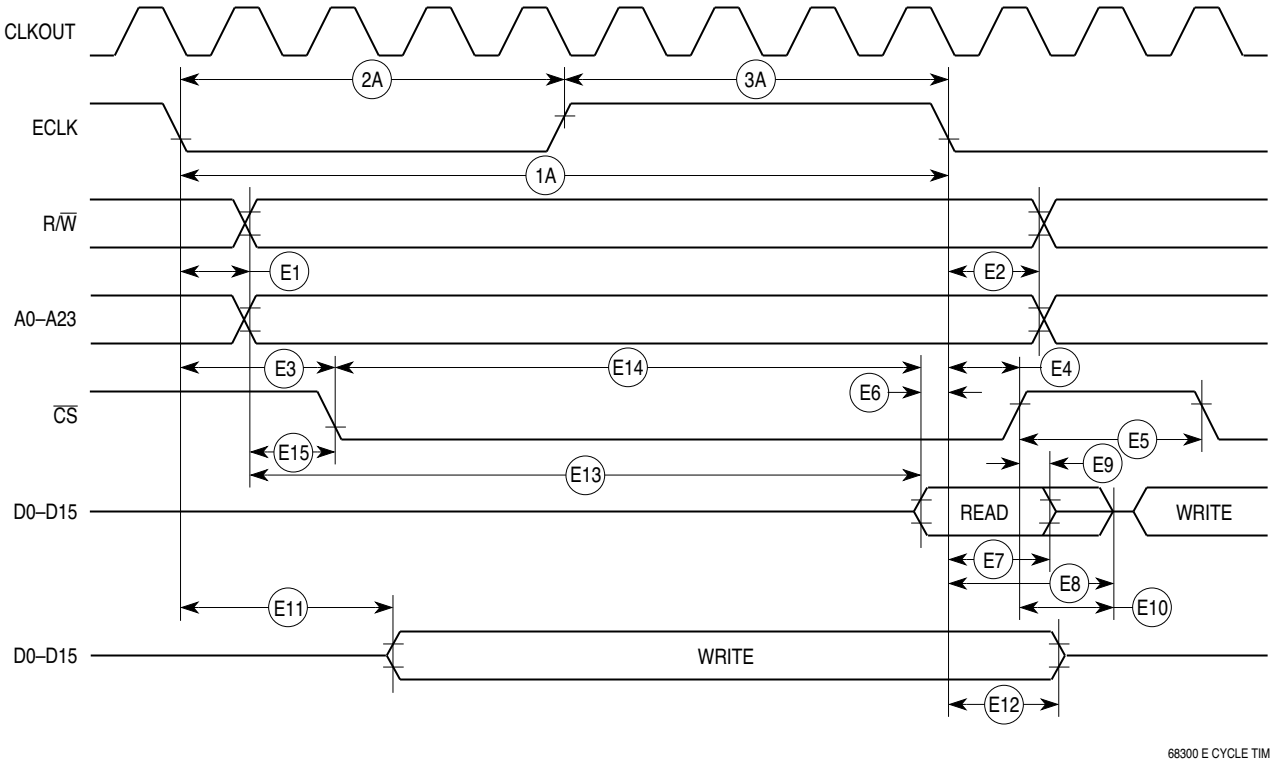
Table A-8a 20.97 MHz ECLK Bus Timing

(V_{DD} = 5.0 Vdc ± 5%, V_{SS} = 0 Vdc, T_A = T_L to T_H)

Num	Characteristic	Symbol	Min	Max	Unit
E1	ECLK Low to Address Valid[2]	t_{EAD}	—	48	ns
E2	ECLK Low to Address Hold	t_{EAH}	10	—	ns
E3	ECLK Low to $\overline{CS}$ Valid ($\overline{CS}$ delay)	t_{ECSD}	—	120	ns
E4	ECLK Low to $\overline{CS}$ Hold	t_{ECSH}	10	—	ns
E5	$\overline{CS}$ Negated Width	t_{ECSN}	25	—	ns
E6	Read Data Setup Time	t_{EDSR}	25	—	ns
E7	Read Data Hold Time	t_{EDHR}	5	—	ns
E8	ECLK Low to Data High Impedance	t_{EDHZ}	—	48	ns
E9	$\overline{CS}$ Negated to Data Hold (Read)	t_{ECDH}	0	—	ns
E10	$\overline{CS}$ Negated to Data High Impedance	t_{ECDZ}	—	1	t_{cyc}
E11	ECLK Low to Data Valid (Write)	t_{EDDW}	—	2	t_{cyc}
E12	ECLK Low to Data Hold (Write)	t_{EDHW}	10	—	ns
E13	Address Access Time (Read)[3]	t_{EACC}	308	—	ns
E14	Chip Select Access Time (Read)[4]	t_{EACS}	236	—	ns
E15	Address Setup Time	t_{EAS}	1/2	—	t_{cyc}

Notes for Tables A–8 and A–8a:

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 = $t_{Ecyc} - t_{EAD} - t_{EDSR}$.
4. Chip select access time = $t_{Ecyc} - t_{ECSD} - t_{EDSR}$.



NOTE: Shown with ECLK = system clock/8 — EDIV bit in clock synthesizer control register (SYNCR) = 0.

Figure A-15 ECLK Timing Diagram

Table A-9 QSPI Timing

(V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = T_L to T_H, 200 pF load on all QSPI pins)

Num	Function	Symbol	Min	Max	Unit
	Operating Frequency Master Slave	f_{op}	 DC DC	 1/4 1/4	 System Clock Frequency System Clock Frequency
1	Cycle Time Master Slave	t_{qcyc}	 4 4	 510 —	 t_{cyc} t_{cyc}
2	Enable Lead Time Master Slave	t_{lead}	 2 2	 128 —	 t_{cyc} t_{cyc}
3	Enable Lag Time Master Slave	t_{lag}	 — 2	 1/2 —	 SCK t_{cyc}
4	Clock (SCK) High or Low Time Master Slave[2]	t_{sw}	 2 t_{cyc} – 60 2 t_{cyc} – n	 255 t_{cyc} —	 ns ns
5	Sequential Transfer Delay Master Slave (Does Not Require Deselect)	t_{td}	 17 13	 8192 —	 t_{cyc} t_{cyc}
6	Data Setup Time (Inputs) Master Slave	t_{su}	 30 20	 — —	 ns ns
7	Data Hold Time (Inputs) Master Slave	t_{hi}	 0 20	 — —	 ns ns
8	Slave Access Time	t_a	—	1	t_{cyc}
9	Slave MISO Disable Time	t_{dis}	—	2	t_{cyc}
10	Data Valid (after SCK Edge) Master Slave	t_v	 — —	 50 50	 ns ns
11	Data Hold Time (Outputs) Master Slave	t_{ho}	 0 0	 — —	 ns ns
12	Rise Time Input[3] Output	 t_{ri} t_{ro}	 — —	 2 30	 µs ns
13	Fall Time Input[3] Output	 t_{fi} t_{fo}	 — —	 2 30	 µs ns

Notes:
1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
2. In formula, n = External SCK rise + 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.

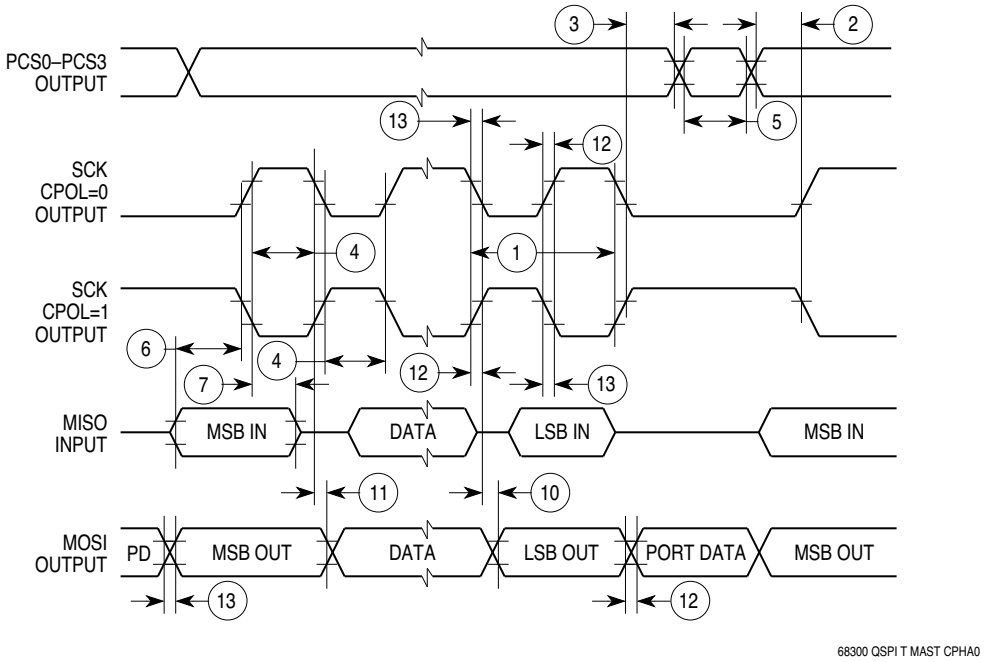


Figure A-16 QSPI Timing — Master, CPHA = 0

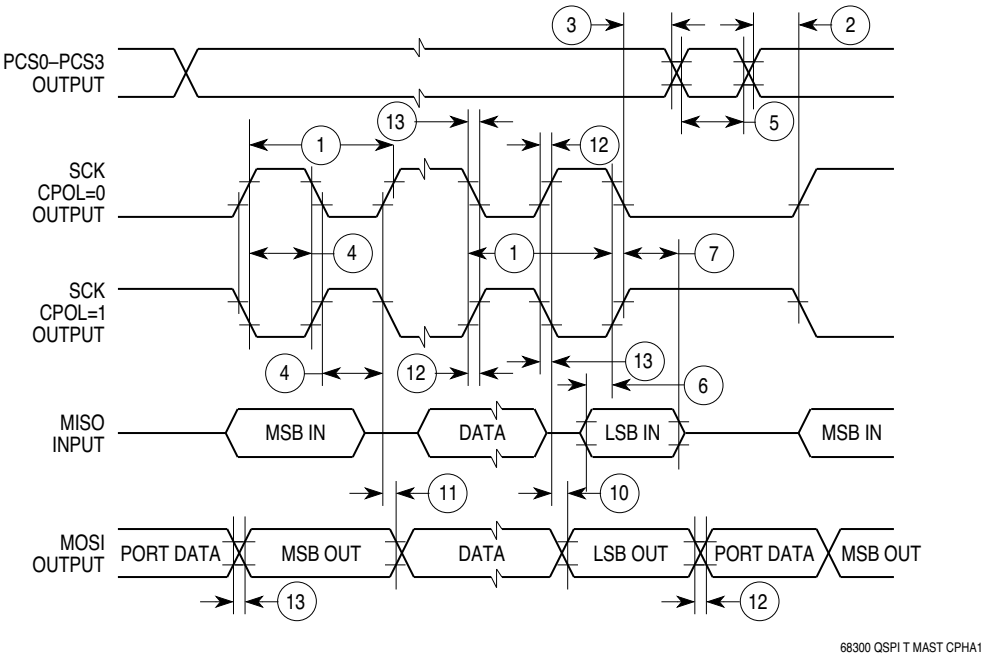


Figure A-17 QSPI Timing — Master, CPHA = 1

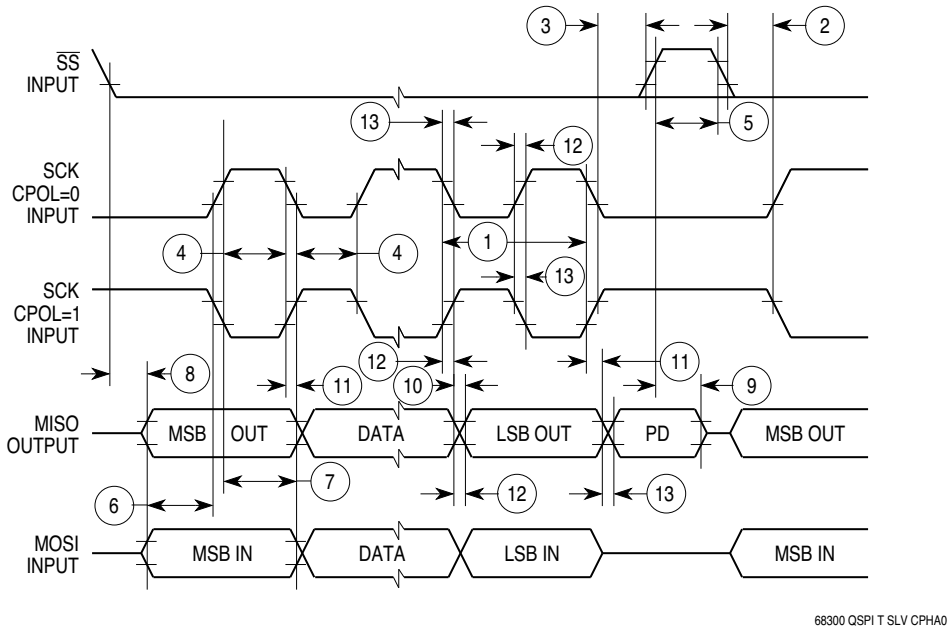


Figure A-18 QSPI Timing — Slave, CPHA = 0

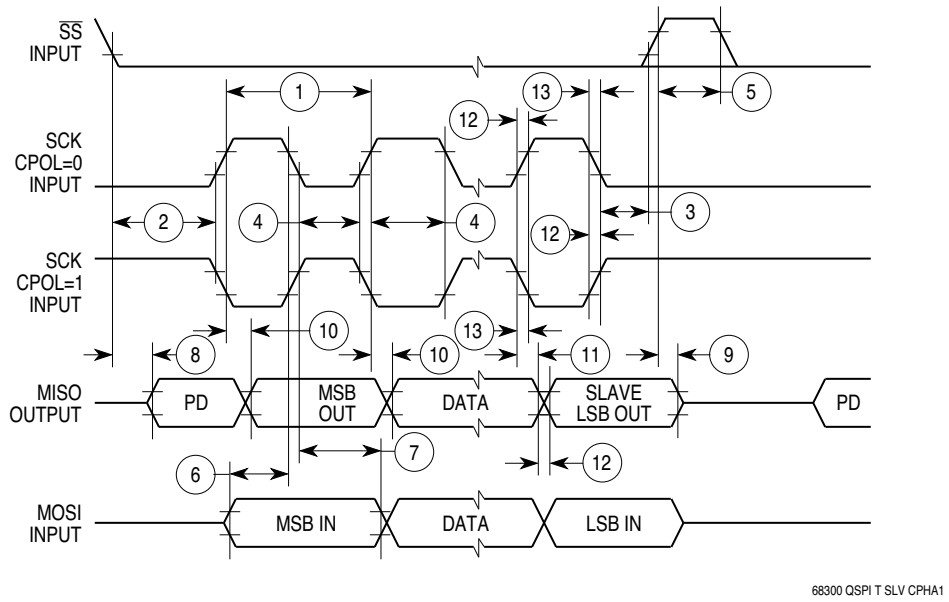


Figure A-19 QSPI Timing — Slave, CPHA = 1

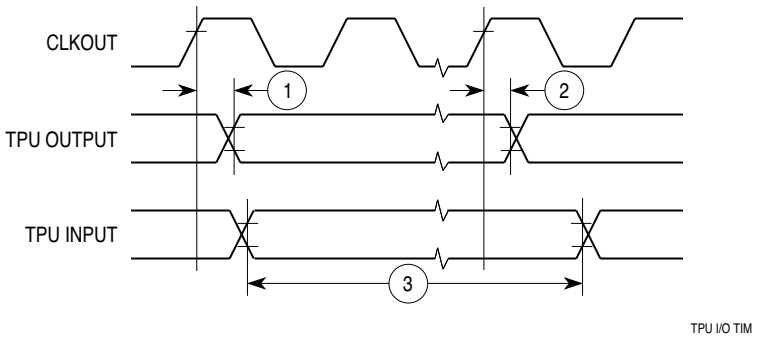


Figure A-20 TPU Timing Diagram

Table A-10 16.78 MHz Time Processor Unit Timing

(V_{DD} and V_{DDSYN} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, $T_A = T_L$ to $T_{H,\ 32.768\ kHz\ reference}$)

Num	Rating	Symbol	Min	Max	Unit
1	CLKOUT High to TPU Output Channel Valid	t_{CHTOV}	2	23	ns
2	CLKOUT High to TPU Output Channel Hold	t_{CHTOH}	0	20	ns
3	TPU Input Channel Pulse Width	t_{TIPW}	4	—	t_{cyc}

Table A-11 20.97 MHz Time Processor Unit Timing

(V_{DD} and V_{DDSYN} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, $T_A = T_L$ to $T_{H,\ 32.768\ kHz\ reference}$)

Num	Rating	Symbol	Min	Max	Unit
1	CLKOUT High to TPU Output Channel Valid	t_{CHTOV}	2	18	ns
2	CLKOUT High to TPU Output Channel Hold	t_{CHTOH}	0	15	ns
3	TPU Input Channel Pulse Width	t_{TIPW}	4	—	t_{cyc}

NOTES:
1.AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels.
2.Timing not valid for external T2CLK input.
3.Maximum load capacitance for CLKOUT pin is 90 pF.
4.Maximum load capacitance for TPU output pins is 100 pF.

APPENDIX B MECHANICAL DATA AND ORDERING INFORMATION

This section contains detailed information to be used as a guide when ordering.

The MC68332 is available in either a 132-pin or 144-pin plastic surface mount package. This appendix provides package pin assignment drawings, dimensional drawings, and ordering information.

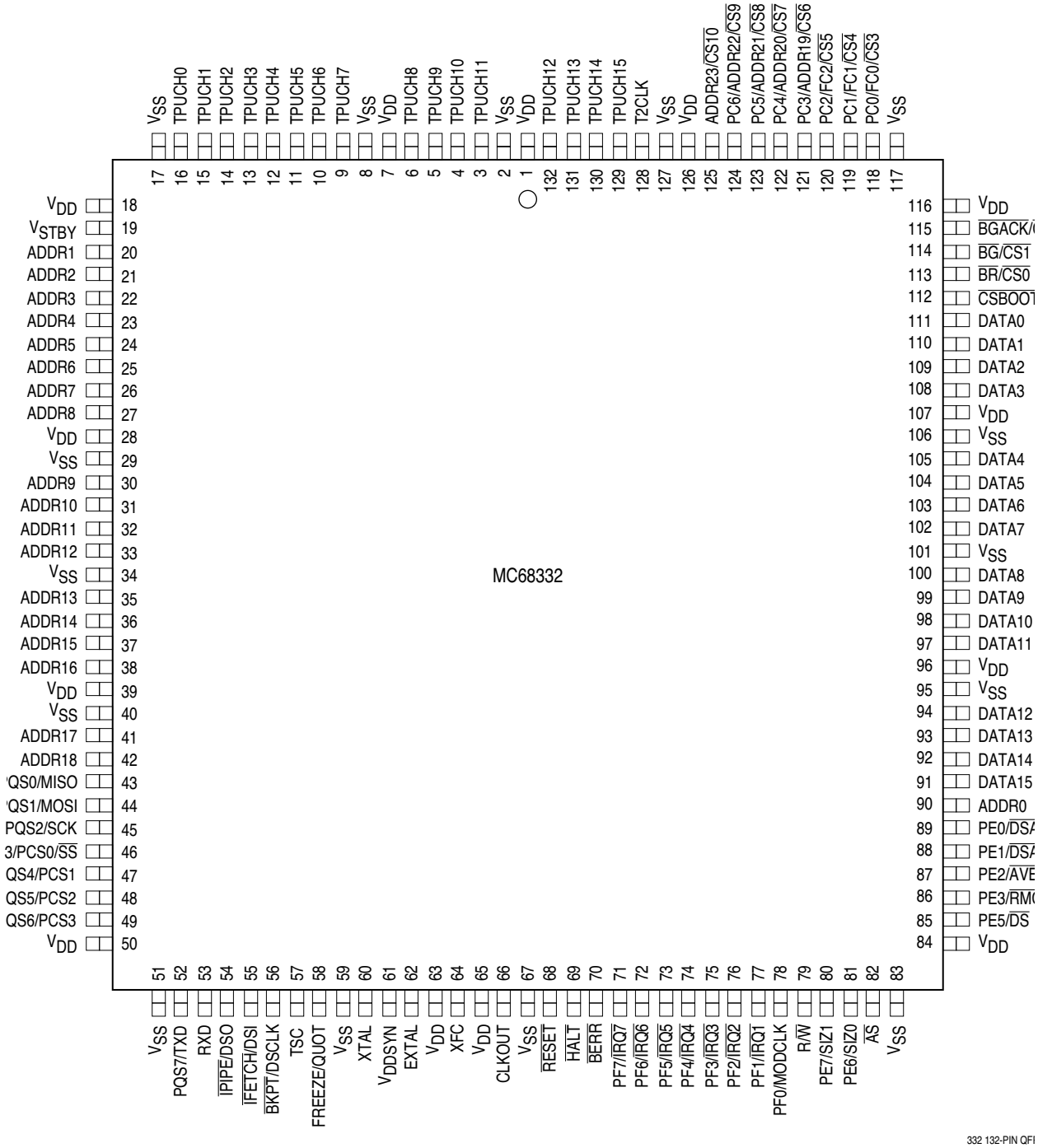
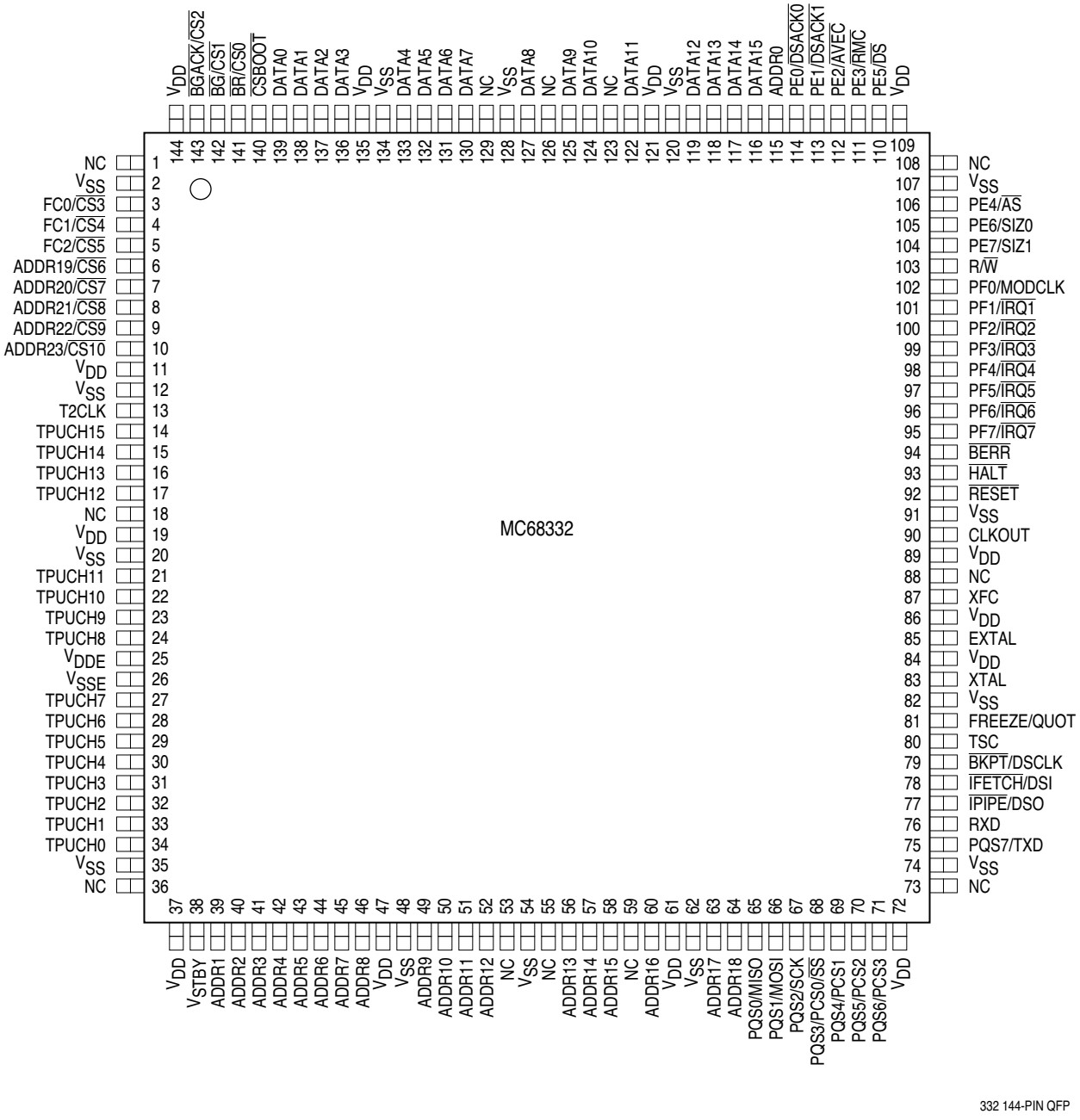


Figure B-1 132-Pin Plastic Surface Mount Package Pin Assignments



332 144-PIN QFP

Figure B-2 144-Pin Plastic Surface Mount Package Pin Assignments

Case outlines number 831A-01 issue A, 863C-01 issue O, and 918-02 issue A are available on the web at

Table B-1 MCU Ordering Information

Package Type	TPU Type	Temperature	Frequency (MHz)	Package Order Quantity	Order Number
132-pin PQFP	Motion Control	–40 to +85 °C	16 MHz	2 pc tray	SPAKMC332GCFC16
				36 pc tray	MC68332GCFC16*
			20 MHz	2 pc tray	SPAKMC332GCFC20
				36 pc tray	MC68332GCFC20*
		–40 to +105 °C	16 MHz	2 pc tray	SPAKMC332GVFC16
				36 pc tray	MC68332GVFC16*
			20 MHz	2 pc tray	SPAKMC332GVFC20
				36 pc tray	MC68332GVFC20*
		–40 to +125 °C	16 MHz	2 pc tray	SPAKMC332GMFC16
				36 pc tray	MC68332GMFC16*
			20 MHz	2 pc tray	SPAKMC332GMFC20
				36 pc tray	MC68332GMFC20*
	Std w/enhanced PPWA	–40 to +85 °C	16 MHz	2 pc tray	SPAKMC332ACFC16
				36 pc tray	MC68332ACFC16*
			20 MHz	2 pc tray	SPAKMC332ACFC20
				36 pc tray	MC68332ACFC20*
		–40 to +105 °C	16 MHz	2 pc tray	SPAKMC332AVFC16
				36 pc tray	MC68332AVFC16*
			20 MHz	2 pc tray	SPAKMC332AVFC20
				36 pc tray	MC68332AVFC20*
		–40 to +125 °C	16 MHz	2 pc tray	SPAKMC332AMFC16
				36 pc tray	MC68332AMFC16*
			20 MHz	2 pc tray	SPAKMC332AMFC20
				36 pc tray	MC68332AMFC20*

Table B-1 MCU Ordering Information (Continued)

Package Type	TPU Type	Temperature	Frequency (MHz)	Package Order Quantity	Order Number
144-pin QFP	Motion Control	–40 to +85 °C	16 MHz	2 pc tray	SPAKMC332GCFV16
				44 pc tray	MC68332GCFV16*
			20 MHz	2 pc tray	SPAKMC332GCFV20
				44 pc tray	MC68332GCFV20*
		–40 to +105 °C	16 MHz	2 pc tray	SPAKMC332GVFV16
				44 pc tray	MC68332GVFV16*
			20 MHz	2 pc tray	SPAKMC332GVFV20
				44 pc tray	MC68332GVFV20*
		–40 to +125 °C	16 MHz	2 pc tray	SPAKMC332GMFV16
				44 pc tray	MC68332GMFV16*
			20 MHz	2 pc tray	SPAKMC332GMFV20
				44 pc tray	MC68332GMFV20*
	Std w/enhanced PPWA	–40 to +85 °C	16 MHz	2 pc tray	SPAKMC332ACFV16
				44 pc tray	MC68332ACFV16*
			20 MHz	2 pc tray	SPAKMC332ACFV20
				44 pc tray	MC68332ACFV20*
		–40 to +105 °C	16 MHz	2 pc tray	SPAKMC332AVFV16
				44 pc tray	MC68332AVFV16*
			20 MHz	2 pc tray	SPAKMC332AVFV20
				44 pc tray	MC68332AVFV20*
		–40 to +125 °C	16 MHz	2 pc tray	SPAKMC332AMFV16
				44 pc tray	MC68332AMFV16*
			20 MHz	2 pc tray	SPAKMC332AMFV20
				44 pc tray	MC68332AMFV20*

Table B-1 MCU Ordering Information (Continued)

Package Type	TPU Type	Temperature	Frequency (MHz)	Package Order Quantity	Order Number
144-pin TQFP	Motion Control	–40 to +85 °C	16 MHz	2 pc tray	SPAKMC332GCPV16
				60 pc tray	MC68332GCPV16*
			20 MHz	2 pc tray	SPAKMC332GCPV20
				60 pc tray	MC68332GCPV20*
		–40 to +105 °C	16 MHz	2 pc tray	SPAKMC332GVPV16
				60 pc tray	MC68332GVPV16*
			20 MHz	2 pc tray	SPAKMC332GVPV20
				60 pc tray	MC68332GVPV20*
		–40 to +125 °C	16 MHz	2 pc tray	SPAKMC332GMPV16
				60 pc tray	MC68332GMPV16*
			20 MHz	2 pc tray	SPAKMC332GMPV20
				60 pc tray	MC68332GMPV20*
	Std w/enhanced PPWA	–40 to +85 °C	16 MHz	2 pc tray	SPAKMC332ACPV16
				60 pc tray	MC68332ACPV16*
			20 MHz	2 pc tray	SPAKMC332ACPV20
				60 pc tray	MC68332ACPV20*
		–40 to +105 °C	16 MHz	2 pc tray	SPAKMC332AVPV16
				60 pc tray	MC68332AVPV16*
			20 MHz	2 pc tray	SPAKMC332AVPV20
				60 pc tray	MC68332AVPV20*
		–40 to +125 °C	16 MHz	2 pc tray	SPAKMC332AMPV16
				60 pc tray	MC68332AMPV16*
			20 MHz	2 pc tray	SPAKMC332AMPV20
				60 pc tray	MC68332AMPV20*

*Quantity orders are designated by a part number suffix (refer to Table B-2). Contact your Freescale representative for more information.

Table B-2 Quantity Order Suffix

FC	FV	PV
36	44	60
180	220	300
360	440	600

APPENDIX C DEVELOPMENT SUPPORT

This section serves as a brief reference to Freescale development tools for the MC68332 microcontroller. Information provided is complete as of the time of publication, but new systems and software are continually being developed. In addition, there is a growing number of third-party tools available. The Freescale *MCU Tool Box* (MCUTLBX/D Rev. C) provides an up-to-date list of development tools. Contact your Freescale representative for further information.

Table C-1 MC68332 Development Tools

Microcontroller Part Number	Modular Development System	Modular Evaluation System
MC68332	M68MMDS1632	M68MEVB1632

C.1 M68MMDS1632 Modular Development System

The M68MMDS1632 Freescale Modular Development System (MMDS) is a development tool for evaluating M68HC16 and M68300 MCU-based systems. The MMDS1632 is an emulator, bus state analyzer, and control station for debugging hardware and software. A separately purchased active probe completes MMDS functionality with regard to a particular MCU or MCU family. The many active probes available let your MMDS emulate a variety of different MCUs. Contact your Freescale sales representative, who will assist you in selecting and configuring the modular system that fits your needs. A full-featured development system, the MMDS provides both in-circuit emulation and bus analysis capabilities, including:

- Real-time in-circuit emulation at maximum speed of 20 MHz (can be upgraded to 33 MHz)
- Built-in emulation memory
  - — 1 Mbyte main emulation memory (fast termination, 2 bus cycle)
  - — 4 Kbytes dual-port emulation memory
- Real-time bus analysis
  - — Instruction disassembly
  - — State-machine-controlled triggering
- Four hardware breakpoints, bitwise masking
- Analog/digital emulation
- Synchronized signal output
- Built-in AC power supply, 85–264 V, 50–60 Hz, FCC and EC EMI compliant
- RS-232 connection to host capable of communicating at 1200, 2400, 4800, 9600, 19200, 38400, or 57600 baud

C.2 M68MEVB1632 Modular Evaluation Board

The M68MEVB1632 Modular Evaluation Board (MEVB) is a development tool for evaluating M68HC16 and M68300 MCU-based systems. The MEVB consists of the M68HC16MPFB modular platform board, an MCU personality board (MPB), an in-circuit debugger printed circuit board (ICD16 or ICD32), and development software. MEVB features include:

- An economical means of evaluating target systems incorporating M68HC16 and M68300 HCMOS MCU devices.
- Expansion memory sockets for installing RAM, EPROM, or EEPROM.
  - — Data RAM: 32K x 16, 128K x 16, or 512K x 16
  - — EPROM/EEPROM: 32K x 16, 64K x 16, 128K x 16, 256K x 16, or 512K x 16
  - — Fast RAM: 32K x 16 or 128K x 16
- Background-mode operation, for detailed operation from a personal computer platform without an on-board monitor.
- Integrated assembly/editing/evaluation/programming environment for easy development.
- As many as seven software breakpoints.
- Re-usable ICD hardware for your target application debug or control.
- Two RS-232C terminal input/output (I/O) ports for user evaluation of the serial communication interface.
- Logic analyzer pod connectors.
- Port replacement unit (PRU) to rebuild I/O ports lost to address/data/control.
- On-board V_{PP} (+12 Vdc) generation for MCU and flash EEPROM programming.
- On-board wire-wrap area.

APPENDIX D REGISTER SUMMARY

This appendix contains MCU address maps, register diagrams, and bit/field definitions. More detailed information about register function is provided in the appropriate sections of the manual.

Except for central processing unit resources, information is presented in the intermodule bus address order shown in **Table D-1**.

Table D-1 Module Address Map

Module	Size (Bytes)	Base Address
SIM	128	$YFFA00
TPURAM CNTL	64	$YFFB00
QSM	512	$YFFC00
TPU	512	$YFFE00

Control registers for all the modules in the microcontroller are mapped into a 4-Kbyte block. The state of the module mapping (MM) bit in the SIM configuration register (SIMCR) determines where the control registers block is located in the system memory map. When MM = 0, register addresses range from $7FF000 to $7FFFFF; when MM = 1, register addresses range from $FFF000 to $FFFFFF.

In the module memory maps in this appendix, the “Access” column specifies which registers are accessible at the supervisor privilege level only and which registers can be assigned to either the supervisor or user privilege level.

D.1 Central Processing Unit

CPU32 registers are not part of the module address map. The following diagram is a functional representation of CPU resources.

D.1.1 CPU32 Register Model

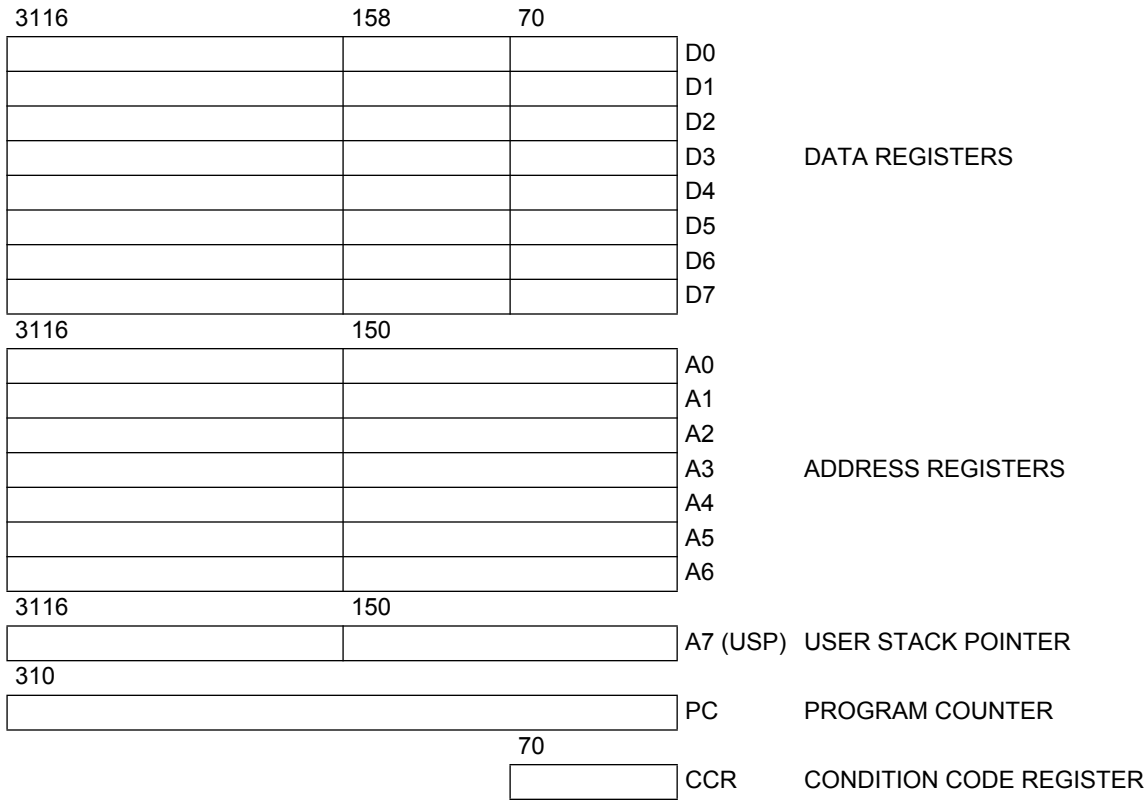


Figure D-1 User Programming Model

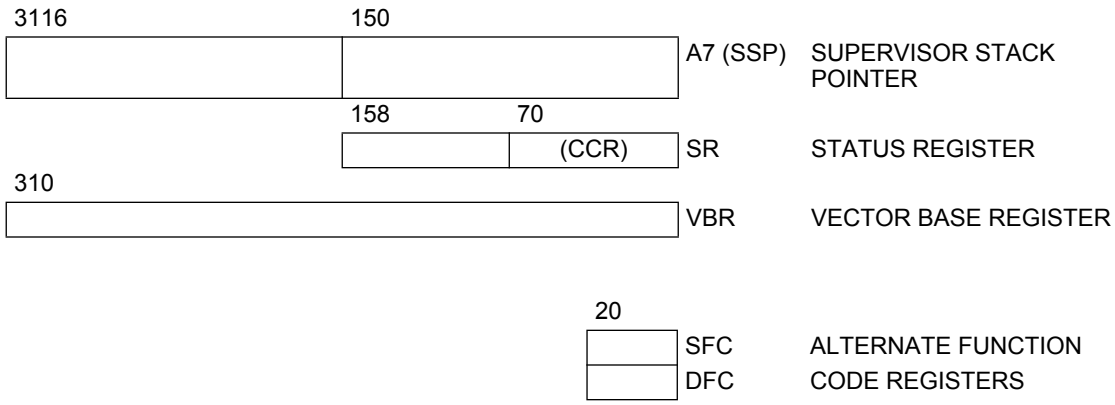


Figure D-2 Supervisor Programming Model Supplement

D.1.2 SR — Status Register

T[1:0]	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

The status register (SR) contains condition codes, an interrupt priority mask, and three control bits. The condition codes are contained in the condition code register (CCR), the lower byte of the SR. (The lower and upper bytes of the status register are also referred to as the user and system bytes, respectively.) At the user privilege level, only the CCR is available. At the supervisor level, software can access the full status register.

T[1:0] — Trace Enable
- 00 = No tracing
- 01 = Trace on change of flow
- 10 = Trace on instruction execution
- 11 = Undefined; reserved

S — Supervisor/User State
- 0 = CPU operates at user privilege level
- 1 = CPU operates at supervisor privilege level

IP[2:0] — Interrupt Priority Mask
The priority value in this field (0 to 7) is used to mask interrupts.

X — Extend Flag
Used in multiple-precision arithmetic operations. In many instructions it is set to the same value as the C bit.

N — Negative Flag
Set when the MSB of a result register is set.

Z — Zero Flag
Set when all bits of a result register are zero.

V — Overflow Flag
Set when two's complement overflow occurs as the result of an operation.

C — Carry Flag
Set when a carry or borrow occurs during an arithmetic operation. Also used during shift and rotate instructions to facilitate multiple word operations.

D.2 System Integration Module

Table D-2 is the SIM address map. The column labeled “Access” indicates the privilege level at which the CPU must be operating to access the register. A designation of “S” indicates that supervisor access is required. A designation of “S/U” indicates that the register can be programmed to the desired privilege level.

Table D-2 SIM Address Map

Access	Address	15 8	7 0
S	$YFFA00	SIM CONFIGURATION (SIMCR)	
S	$YFFA02	FACTORY TEST (SIMTR)	
S	$YFFA04	CLOCK SYNTHESIZER CONTROL (SYNCR)	
S	$YFFA06	NOT USED	RESET STATUS REGISTER (RSR)
S	$YFFA08	MODULE TEST E (SIMTRE)	
S	$YFFA0A	NOT USED	NOT USED
S	$YFFA0C	NOT USED	NOT USED
S	$YFFA0E	NOT USED	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 (PICR)	
S	$YFFA24	PERIODIC INTERRUPT TIMING (PITR)	
S	$YFFA26	NOT USED	SOFTWARE SERVICE (SWSR)
S	$YFFA28	NOT USED	NOT USED
S	$YFFA2A	NOT USED	NOT USED
S	$YFFA2C	NOT USED	NOT USED
S	$YFFA2E	NOT USED	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	NOT USED
	$YFFA3E	NOT USED	NOT USED
S/U	$YFFA40	NOT USED	PORT C DATA (PORTC)
	$YFFA42	NOT USED	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)	
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)	
	$YFFA5C	CHIP-SELECT BASE 4 (CSBAR4)	

Table D-2 SIM Address Map

Access	Address	15 8	7 0
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	NOT USED
	$YFFA7A	NOT USED	NOT USED
	$YFFA7C	NOT USED	NOT USED
	$YFFA7E	NOT USED	NOT USED

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

D.2.1 SIMCR — Module Configuration Register $YFFA00

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

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

EXOFF — External Clock Off

0 = The CLKOUT pin is driven from an internal clock source.
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 during software debug.

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

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.

SUPV — Supervisor/Unrestricted Data Space

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

0 = Registers with access controlled by the SUPV bit are accessible from either the user or supervisor privilege level.

1 = Registers with access controlled by the SUPV bit are restricted to supervisor access only.

MM — Module Mapping

0 = Internal modules are addressed from $7FF000 – $7FFFFF.

1 = Internal modules are addressed from $FFF000 – $FFFFFF.

IARB[3:0] — Interrupt Arbitration Field

Determines SIM interrupt arbitration priority. The reset value is $F (highest priority), to prevent SIM interrupts from being discarded during initialization.

D.2.2 SIMTR — System Integration Test Register $YFFA02

SIMTR is used for factory test only.

D.2.3 SYNCR — Clock Synthesizer Control Register $YFFA04

15	14	13 – 8	7	6	5	4	3	2	1	0
W	X	Y	EDIV	0	0	SLIMP	SLOCK	RSTEN	STSIM	STEXT
RESET:										
0	0	1 1 1 1 1 1	0	0	0	U	U	0	0	0

SYNCR determines system clock operating frequency and mode of operation. Clock frequency is determined by SYNCR bit settings as follows:

W — Frequency Control (VCO)

0 = Base VCO frequency

1 = VCO frequency multiplied by four

X — Frequency Control Bit (Prescale)

0 = VCO frequency divided by four (base system clock frequency)

1 = VCO frequency divided by two (system clock frequency doubles)

Y[5:0] — Frequency Control (Counter)

The Y field is the initial value for the modulus 64 down counter in the synthesizer feedback loop. Values range from 0 to 63.

EDIV — ECLK Divide Rate

0 = ECLK is system clock divided by 8

1 = ECLK is system clock divided by 16

SLIMP — Limp Mode

0 = External crystal is VCO reference

1 = Loss of crystal reference

SLOCK — Synthesizer Lock

0 = VCO is enabled, but has not locked.

1 = VCO has locked on the desired frequency or system clock is external.

RSTEN — Reset Enable
0 = Loss of reference causes the MCU to operate in limp mode.
1 = Loss of reference causes system reset.

STSIM — Stop Mode System Integration Clock
0 = SIM clock driven by an external source and VCO off during low-power stop.
1 = SIM clock driven by VCO during low-power stop.

STEXT — Stop Mode External Clock
0 = CLKOUT held low during low-power stop.
1 = CLKOUT driven from SIM clock during low-power stop.

D.2.4 RSR — Reset Status Register $YFFA07

15 … 8	7	6	5	4	3	2	1	0
NOT USED	EXT	POW	SW	HLT	0	LOC	SYS	TST

RSR contains a status bit for each reset source in the MCU. RSR is updated when the MCU comes out of reset. A set bit indicates what type of reset occurred. If multiple sources assert reset signals at the same time, more than one bit in RSR may be set. This register can be read at any time; a write has no effect.

EXT — External Reset
Reset caused by an external signal.

POW — Power-Up Reset
Reset caused by the power-up reset circuit.

SW — Software Watchdog Reset
Reset caused by the software watchdog circuit.

HLT — Halt Monitor Reset
Reset caused by the halt monitor.

LOC — Loss of Clock Reset
Reset caused by loss of clock frequency reference.

SYS — System Reset
Reset caused by a RESET instruction.

TST — Test Submodule Reset
Reset caused by the test submodule. Used during system test only.

D.2.5 SIMTRE — System Integration Test Register (ECLK) $YFFA08

Register is used for factory test only.

D.2.6 PORTE0/PORTE1 — Port E Data Register $YFFA11, $YFFA13

15 … 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

PORTE is an internal data latch that can be accessed at two locations. PORTE can be read or written at any time. If a pin in I/O port E is configured as an output, the corresponding bit value is driven out on the pin. When a pin is configured for output, a read of PORTE returns the latched bit value; when a pin is configured for input, a read returns the pin logic level.

D.2.7 DDRE — Port E Data Direction Register $YFFA15

15 … 8	7	6	5	4	3	2	1	0
NOT USED	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0
RESET:	0	0	0	0	0	0	0	0

Bits in this register control the direction of the port E pin drivers when pins are configured for I/O. Setting a bit configures the corresponding pin as an output; clearing a bit configures the corresponding pin as an input. This register can be read or written at any time.

D.2.8 PEPAR — Port E Pin Assignment Register $YFFA17

15 … 8	7	6	5	4	3	2	1	0
NOT USED	PEPA7	PEPA6	PEPA5	PEPA4	PEPA3	PEPA2	PEPA1	PEPA0
RESET:	DATA8	DATA8	DATA8	DATA8	DATA8	DATA8	DATA8	DATA8

Bits in this register determine the function of port E pins. Setting a bit assigns the corresponding pin to a bus control signal; clearing a bit assigns the pin to I/O port E.

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	$\overline{DSACK0}$

D.2.9 PORTF0/PORTF1 — Port F Data Register $YFFA19, $YFFA1B

15 … 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

PORTF is an internal data latch that can be accessed at two locations. It can be read or written at any time. If a pin in I/O port F is configured as an output, the corresponding bit value is driven out on the pin. When a pin is configured for output, a read of PORTF returns the latched bit value; when a pin is configured for input, a read returns the pin logic level.

D.2.10 DDRF — Port F Data Direction Register $YFFA1D

15 … 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

Bits in this register control the direction of the port F pin drivers when pins are configured for I/O. Setting a bit configures the corresponding pin as an output; clearing a bit configures the corresponding pin as an input. This register can be read or written at any time.

D.2.11 PFPAR — Port F Pin Assignment Register $YFFA1F

15 … 8	7	6	5	4	3	2	1	0
NOT USED	PFPA7	PFPA6	PFPA5	PFPA4	PFPA3	PFPA2	PFPA1	PFPA0
RESET:	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9

Bits in this register determine the function of port F pins. Setting a bit assigns the corresponding pin to a control signal; clearing a bit assigns the pin to port F.

Port F Pin Assignments		
PFPAR Field	**Port F Signal**	**Control 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

D.2.12 SYPCR — System Protection Control Register $YFFA21

15	8	7	6	5	4	3	2	1	0
NOT USED		SWE	SWP	SWT		HME	BME	BMT	
RESET:		1	MODCLK	0	0	0	0	0	0

SYPCR controls system monitor functions, software watchdog clock prescaling, and bus monitor timing. This register can be written once following power-on or reset.

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

SWP — Software Watchdog Prescale
- 0 = Software watchdog clock not prescaled
- 1 = Software watchdog clock prescaled by 512

SWT[1:0] — Software Watchdog Timing
This field selects software watchdog time-out period.

Software Watchdog Ratio		
SWP	**SWT**	**Ratio**
0	00	2^9
0	01	2^{11}
0	10	2^{13}
0	11	2^{15}
1	00	2^{18}
1	01	2^{20}
1	10	2^{22}
1	11	2^{24}

HME — Halt Monitor Enable
- 0 = Disable halt monitor function
- 1 = Enable halt monitor function

BME — Bus Monitor External Enable
- 0 = Disable bus monitor function for an internal to external bus cycle.
- 1 = Enable bus monitor function for an internal to external bus cycle.

BMT[1:0] — Bus Monitor Timing
This field selects bus monitor time-out period.

Bus Monitor Period	
BMT	**Bus Monitor Time-out Period**
00	64 System Clocks
01	32 System Clocks
10	16 System Clocks
11	8 System Clocks

D.2.13 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			PIV							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

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

PIRQL[2:0] — Periodic Interrupt Request Level
This field determines the priority of periodic interrupt requests.

PIV[7:0] — Periodic Interrupt Vector
The bits of this field contain the interrupt vector number supplied by the SIM when the CPU acknowledges an interrupt request.

D.2.14 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							
RESET:															
0	0	0	0	0	0	0	$\overline{\text{MODCLK}}$	0	0	0	0	0	0	0	0

Contains the count value for 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 a value of 512

PITM[7:0] — Periodic Interrupt Timing Modulus
This is the 8-bit timing modulus used to determine periodic interrupt rate. Use the following expression to calculate timer period.

D.2.15 SWSR — Software Service Register $YFFA27

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

When the software watchdog is enabled, a service sequence must be written to this register within a specific interval. When read, SWSR always returns $00. Register shown with read value.

D.2.16 TSTMSRA — Master Shift Register A $YFFA30

Register is used for factory test only.

D.2.17 TSTMSRB — Master Shift Register B $YFFA32

Register is used for factory test only.

D.2.18 TSTSC — Test Module Shift Count $YFFA34

Register is used for factory test only.

D.2.19 TSTRC — Test Module Repetition Count $YFFA36

Register is used for factory test only.

D.2.20 CREG — Test Submodule Control Register $YFFA38

Register is used for factory test only.

D.2.21 DREG — Distributed Register $YFFA3A

Register is used for factory test only.

D.2.22 PORTC — Port C Data Register $YFFA41

15 … 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

PORTC latches data for chip-select pins that are used for discrete output.

D.2.23 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	CSPA0[6]		CSPA0[5]		CSPA0[4]		CSPA0[3]		CSPA0[2]		CSPA0[1]		$\overline{\text{CSBOOT}}$	
RESET:															
0	0	DATA2	1	DATA2	1	DATA2	1	DATA1	1	DATA1	1	DATA1	1	1	DATA0

CSPAR0 Pin Assignments			
CSPAR0 Field	**CSPAR0 Signal**	**Alternate Signal**	**Discrete Output**
CSPA0[6]	$\overline{\text{CS5}}$	FC2	PC2
CSPA0[5]	$\overline{\text{CS4}}$	FC1	PC1
CSPA0[4]	$\overline{\text{CS3}}$	FC0	PC0
CSPA0[3]	$\overline{\text{CS2}}$	$\overline{\text{BGACK}}$	—
CSPA0[2]	$\overline{\text{CS1}}$	$\overline{\text{BG}}$	—
CSPA0[1]	$\overline{\text{CS0}}$	$\overline{\text{BR}}$	—
$\overline{\text{CSBOOT}}$	$\overline{\text{CSBOOT}}$	—	—

Contains seven 2-bit fields, CSPA0[6:1] and $\overline{\text{CSBOOT}}$, that determine the functions of corresponding chip-select pins. CSPAR0[15:14] are not used. These bits always read zero; write has no effect. CSPAR0 bit 1 always reads one; writes to CSPAR0 bit 1 have no effect. The alternate functions can be enabled by data bus mode selection during reset.

D.2.24 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	CSPA1[4]		CSPA1[3]		CSPA1[2]		CSPA1[1]		CSPA1[0]	
RESET:															
0	0	0	0	0	0	DATA7	1	DATA6	1	DATA5	1	DATA4	1	DATA3	1

CSPAR1 Pin Assignments			
CSPAR1 Field	**CSPAR1 Signal**	**Alternate Signal**	**Discrete Output**
CSPA1[4]	$\overline{CS10}$	ADDR23	ECLK
CSPA1[3]	$\overline{CS9}$	ADDR22	PC6
CSPA1[2]	$\overline{CS8}$	ADDR21	PC5
CSPA1[1]	$\overline{CS7}$	ADDR20	PC4
CSPA1[0]	$\overline{CS6}$	ADDR19	PC3

Contains five 2-bit fields (CSPA1[4:0]) that determine the functions of corresponding chip-select pins. CSPAR1[15:10] are not used. These bits always read zero; write has no effect. The CSPAR1 pin assignments table shows alternate functions that can be enabled by data bus mode selection during reset.

Pin Assignment Field Encoding	
Bit Field	**Description**
00	Discrete Output*
01	Alternate Function*
10	Chip Select (8-Bit Port)
11	Chip Select (16-Bit Port)

*Does not apply to the $\overline{CSBOOT}$ field

D.2.25 CSBARBT — Chip Select Base Address Register Boot ROM $YFFA48

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

D.2.26 CSBAR[0:10] — Chip Select Base Address Registers $YFFA4C–$YFFA74

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

Each chip-select pin has an associated base address register. A base address is the lowest address in the block of addresses enabled by a chip select. CSBARBT contains the base address for selection of a bootstrap peripheral memory device. Bit and field definition for CSBARBT and CSBAR[0:10] are the same, but reset block sizes differ.

ADDR[23:11] — Base Address

This field sets the starting address of a particular address space.

BLKSZ — Block Size

This field determines the size of the block above the base address that is enabled by the chip select.

Block Size Encoding		
BLKSZ[2:0]	**Block Size**	**Address Lines Compared**
000	2 K	ADDR[23:11]
001	8 K	ADDR[23:13]
010	16 K	ADDR[23:14]
011	64 K	ADDR[23:16]
100	128 K	ADDR[23:17]
101	256 K	ADDR[23:18]
110	512 K	ADDR[23:19]
111	1 M	ADDR[23:20]

D.2.27 CSORBT — Chip Select Option Register Boot ROM $YFFA4A

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

D.2.28 CSOR[0:10] — Chip Select Option Registers $YFFA4E–$YFFA76

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

Contain parameters that support bootstrap operations from peripheral memory devices. Bit and field definitions for CSORBT and CSOR[0:10] are the same.

MODE — Asynchronous Bus/Synchronous E-Clock Mode

Synchronous mode cannot be used with internally generated autovectors.

0 = Asynchronous mode selected
1 = Synchronous mode selected

BYTE — Upper/Lower Byte Option

The value in this field determines whether a select signal can be asserted.

R/$\overline{W}$ — Read/Write

This field causes a chip select to be asserted only for a read, only for a write, or for both read and write.

STRB — Address Strobe/Data Strobe

0 = Address strobe
1 = Data strobe

$\overline{DSACK}$ — Data Strobe Acknowledge

This field specifies the source of $\overline{DSACK}$ in asynchronous bus mode and controls wait state insertion.

SPACE — Address Space Select

This field selects an address space to be used by the chip-select logic.

IPL — Interrupt Priority Level

This field determines interrupt priority level when a chip select is used for interrupt acknowledge. It does not affect CPU interrupt recognition.

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

Do not enable autovector support when in synchronous mode.

0 = External interrupt vector enabled

1 = Autovector enabled

Option Register Function Summary							
MODE	**BYTE**	**R/$\overline{\text{W}}$**	**STRB**	**$\overline{\text{DSACK}}$**	**SPACE**	**IPL**	**$\overline{\text{AVEC}}$**
0 = ASYNC	00 = Disable	00 = Rsvd	0 = $\overline{\text{AS}}$	0000 = 0 WAIT	00 = CPU SP	000 = All	0 = Off
1 = SYNC	01 = Lower	01 = Read	1 = $\overline{\text{DS}}$	0001 = 1 WAIT	01 = User SP	001 = Priority 1	1 = On
	10 = Upper	10 = Write		0010 = 2 WAIT	10 = Supv SP	010 = Priority 2	
	11 = Both	11 = Both		0011 = 3 WAIT	11 = S/U SP	011 = Priority 3	
				0100 = 4 WAIT		100 = Priority 4	
				0101 = 5 WAIT		101 = Priority 5	
				0110 = 6 WAIT		110 = Priority 6	
				0111 = 7 WAIT		111 = Priority 7	
				1000 = 8 WAIT			
				1001 = 9 WAIT			
				1010 = 10 WAIT			
				1011 = 11 WAIT			
				1100 = 12 WAIT			
				1101 = 13 WAIT			
				1110 = F term			
				1111 = External			

D.3 Standby RAM Module with TPU Emulation

Table D-3 is the TPURAM address map. TPURAM responds to both program and data space accesses. The RASP bit in the TRAMMCR determines whether the processor must be operating at the supervisor privilege level to access the array. TPURAM control registers are accessible at the supervisor privilege level only.

Table D-3 TPURAM Address Map

Access	Address	15 8	7 0
S	$YFFB00	TPURAM MODULE CONFIGURATION REGISTER (TRAMMCR)	
S	$YFFB02	TPURAM TEST REGISTER (TRAMTST)	
S	$YFFB04	TPURAM BASE ADDRESS AND STATUS REGISTER (TRAMBAR)	
S	$YFFB06	NOT USED	

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

D.3.1 TRAMMCR — TPURAM Module Configuration Register $YFFB00

15	14	13	12	11	10	9	8	7 0
STOP	0	0	0	0	0	0	RASP	NOT USED
RESET:								
0	0	0	0	0	0	0	1	

STOP — Stop Control

0 = TPURAM array operates normally.

1 = TPURAM array enters low-power stop mode.

This bit controls whether the RAM array is in stop mode or normal operation. Reset state is zero, for normal operation. In stop mode, the array retains its contents, but cannot be read or written by the CPU.

RASP[1:0] — TPURAM Array Space Field

0 = TPURAM array is accessible from the supervisor or user privilege level.

1 = TPURAM array is accessible from the supervisor privilege level only.

D.3.2 TRAMTST — TPURAM Test Register $YFFB02

TRAMTST is used for factory test of the TPURAM module.

D.3.3 TRAMBAR — TPURAM Base Address and Status Register $YFFB04

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	NOT USED	RAMD S
RESET:														
0	0	0	0	0	0	0	0	0	0	0	0	0		0

ADDR[23:11] — TPURAM Array Base Address

These bits specify address lines ADDR[23:11] of the base address of the TPURAM array when enabled.

RAMDS — RAM Array disabled

0 = RAM array is enabled

1 = RAM array is disabled

The TPURAM array is disabled by internal logic after a master reset. Writing a valid base address to the RAM array base address field (bits [15:3]) automatically clears RAMDS, enabling the RAM array.

D.4 Queued Serial Module

Table D-4 is the QSM address map. The column labeled “Access” indicates the privilege level at which the CPU must be operating to access the register. A designation of “S” indicates that supervisor access is required: a designation of “S/U” indicates that the register can be programmed to the desired privilege level.

Table D-4 QSM Address Map

Access	Address	15 8	7 0
S	$YFFC00	QSM MODULE CONFIGURATION (QSMCR)	
S	$YFFC02	QSM TEST (QTEST)	
S	$YFFC04	QSM INTERRUPT LEVEL (QILR)	QSM INTERRUPT VECTOR (QIVR)
S/U	$YFFC06	NOT USED	
S/U	$YFFC08	SCI CONTROL 0 (SCCR0)	
S/U	$YFFC0A	SCI CONTROL 1 (SCCR1)	
S/U	$YFFC0C	SCI STATUS (SCSR)	
S/U	$YFFC0E	SCI DATA (SCDR)	
S/U	$YFFC10	NOT USED	
S/U	$YFFC12	NOT USED	
S/U	$YFFC14	NOT USED	PQS DATA (PORTQS)
S/U	$YFFC16	PQS PIN ASSIGNMENT (PQSPAR)	PQS DATA DIRECTION (DDRQS)
S/U	$YFFC18	SPI CONTROL 0 (SPCR0)	
S/U	$YFFC1A	SPI CONTROL 1 (SPCR1)	
S/U	$YFFC1C	SPI CONTROL 2 (SPCR2)	
S/U	$YFFC1E	SPI CONTROL 3 (SPCR3)	SPI STATUS (SPSR)
S/U	$YFFC20–$YFFCFF	NOT USED	
S/U QUEUE RAM	$YFFD00–$YFFD1F	RECEIVE RAM (RR[0:F])	
S/U QUEUE RAM	$YFFD20–$YFFD3F	TRANSMIT RAM (TR[0:F])	
S/U QUEUE RAM	$YFFD40–$YFFD4F	COMMAND RAM (CR[0:F])	

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

D.4.1 QSMCR — QSM Configuration Register $YFFC00

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

QSMCR bits enable stop and freeze modes, and determine the arbitration priority of QSM interrupt requests.

STOP — Stop Enable

0 = Normal QSM clock operation

1 = QSM clock operation stopped

When STOP is set, the QSM enters low-power stop mode. System clock input to the module is disabled. While STOP is asserted, only QSMCR reads are guaranteed to be valid, but writes to QSPI RAM or any register are guaranteed valid. STOP is set during

reset. The SCI receiver and transmitter must be disabled before STOP is set. To stop the QSPI, set the HALT bit in SPCR3, wait until the HALTA flag is set, then set STOP.

FRZ[1:0] — Freeze Control

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 background mode. FRZ0 is reserved for future use.

SUPV — Supervisor/Unrestricted

0 = Supervisor access

1 = User access

IARB — Interrupt Arbitration

Each module that generates interrupts must have an IARB value. IARB values are used to arbitrate between interrupt requests of the same priority.

D.4.2 QTEST — QSM Test Register $YFFC02

Used for factory test only.

D.4.3 QILR — QSM Interrupt Level Register $YFFC04
QIVR — QSM Interrupt Vector Register $YFFC05

15	14	13 11	10 8	7 0
0	0	ILQSPI	ILSCI	INTV
RESET:				
0	0	0 0 0	0 0 0	0 0 0 0

The values of the ILQSPI and ILSCI fields in QILR determine the priority of QSPI and SCI interrupt requests. QIVR determines the value of the interrupt vector number the QSM supplies when it responds to an interrupt acknowledge cycle. At reset, QIVR is initialized to vector number $0F, the uninitialized interrupt vector number. To use interrupt-driven serial communication, a user-defined vector number must be written to QIVR.

ILQSPI — Interrupt Level for QSPI

When an interrupt request is made, ILQSPI value determines which of the interrupt request signals is asserted; when a request is acknowledged, the QSM compares this value to a mask value supplied by the CPU32 to determine whether to respond. ILQSPI must have a value in the range $0 (lowest priority) to $7 (highest priority).

ILSCI — Interrupt Level for SCI

When an interrupt request is made, ILSCI value determines which of the interrupt request signals is asserted. When a request is acknowledged, the QSM compares this value to a mask value supplied by the CPU32 to determine whether to respond. The field must have a value in the range $0 (lowest priority) to $7 (highest priority).

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

INTV[7:0] — Interrupt Vector Number

The values of INTV[7:1] are the same for both QSPI and SCI interrupt requests; the value of INTV0 used during an interrupt acknowledge cycle is supplied by the QSM. INTV0 is at logic level zero during an SCI interrupt and at logic level one during a QSPI interrupt. A write to INTV0 has no effect. Reads of INTV0 return a value of one.

D.4.4 SCCR0 — SCI Control Register 0 $YFFC08

15	14	13	12												0
0	0	0	SCBR												
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

SCCR0 contains the SCI baud rate selection field. Baud rate must be set before the SCI is enabled. The CPU32 can read and write SCCR0 at any time. Changing the value of SCCR0 bits during a transfer operation can disrupt operation.

SCBR — SCI Baud Rate

SCI baud rate is programmed by writing a 13-bit value to this field. Writing a value of zero to SCBR disables the baud rate generator. Baud clock rate is calculated as follows:

$$\text{SCI Baud Clock Rate} = \frac{\text{System Clock}}{32 \times \text{SCBR}}$$

D.4.5 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, including transmitter and receiver enable bits, interrupt enable bits, and operating mode enable bits. The CPU can read and write SCCR1 at any time. The SCI can modify the RWU bit under certain circumstances. Changing the value of SCCR1 bits during a transfer operation can disrupt operation.

LOOPS — Loop Mode

0 = Normal SCI operation, no looping, feedback path disabled
1 = Test SCI operation, looping, feedback path enabled

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.

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

PE — Parity Enable
0 = SCI parity disabled
1 = SCI parity enabled

M — Mode Select
0 = 10-bit SCI frame
1 = 11-bit SCI frame

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 and OR interrupts inhibited
1 = SCI RDRF and OR interrupts 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 can be used as I/O)
1 = SCI transmitter enabled (TXD pin dedicated to SCI transmitter)

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)

SBK — Send Break
0 = Normal operation
1 = Break frame(s) transmitted after completion of current frame

D.4.6 SCSR — SCI Status Register $YFFC0C

15 … 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

SCSR contains flags that show SCI operating conditions. These flags are cleared either by SCI hardware or by a CPU32 read/write sequence. The sequence consists of reading SCSR, then reading or writing SCDR.

If an internal SCI signal for setting a status bit comes after the CPU32 has read the asserted status bits, but before the CPU has written or read SCDR, the newly set status bit is not cleared. SCSR must be read again with the bit set and SCDR must be written or read before the status bit is cleared.

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

TDRE — Transmit Data Register Empty
- 0 = Register TDR still contains data to be sent to the transmit serial shifter.
- 1 = A new character can now be written to register TDR.

TC — Transmit Complete
- 0 = SCI transmitter is busy.
- 1 = SCI transmitter is idle.

RDRF — Receive Data Register Full
- 0 = Register RDR is empty or contains previously read data.
- 1 = Register RDR contains new data.

RAF — Receiver Active
- 0 = SCI receiver is idle.
- 1 = SCI receiver is busy.

IDLE — Idle-Line Detected
- 0 = SCI receiver did not detect an idle-line condition.
- 1 = SCI receiver detected an idle-line condition.

OR — Overrun Error
- 0 = RDRF is cleared before new data arrives.
- 1 = RDRF is not cleared before new data arrives.

NF — Noise Error Flag
- 0 = No noise detected on the received data
- 1 = Noise occurred on the received data.

FE — Framing Error
- 0 = No framing error on the received data
- 1 = Framing error or break occurred on the received data.

PF — Parity Error
0 = No parity error on the received data
1 = Parity error occurred on the received data.

D.4.7 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 consists of two data registers located at the same address. RDR is a read-only register that contains data received by the SCI serial interface. Data comes into the receive serial shifter and is transferred to RDR. TDR is a write-only register that contains data to be transmitted. Data is first written to TDR, 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 nine-bit operation. When the SCI is configured for eight-bit operation, R8/T8 have no meaning or effect.

D.4.8 PORTQS — Port QS Data Register $YFFC15

15	8	7	6	5	4	3	2	1	0
NOT USED		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.

D.4.9 PQSPAR — PORT QS Pin Assignment Register $YFFC16
DDRQS — PORT QS Data Direction Register $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 PQSPAR assigns the corresponding pin to general-purpose I/O; setting a bit assigns the pin to the QSPI. PQSPAR does not affect operation of the SCI.

PQSPAR Pin Assignments		
PQSPAR Field	**PQSPAR Bit**	**Pin Function**
PQSPA0	0	PQS0
	1	MISO
PQSPA1	0	PQS1
	1	MOSI
PQSPA2	0	PQS2[1]
	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[2]
	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 set), 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.

Effect of DDRQS on PORTQS Pins		
Pin	**DDRQS Bit**	**Pin Function**
PQS0	0	Digital Input
	1	Digital Output
PQS1	0	Digital Input
	1	Digital Output
PQS2	0	Digital Input
	1	Digital Output
PQS2	0	Digital Input
	1	Digital Output
PQS3	0	Digital Input
	1	Digital Output
PQS4	0	Digital Input
	1	Digital Output
PQS5	0	Digital Input
	1	Digital Output
PQS6	0	Digital Input
	1	Digital Output
PQS7	0	Digital Input
	1	Digital Output

Effect of DDRQS on QSM Pin Function				
QSM Pin	**Mode**	**DDRQS Bit**	**Bit State**	**Pin Function**
MISO	Master	DDQS0	0	Serial Data Input to QSPI
			1	Disables Data Input
	Slave		0	Disables Data Output
			1	Serial Data Output from QSPI
MOSI	Master	DDQS1	0	Disables Data Output
			1	Serial Data Output from QSPI
	Slave		0	Serial Data Input to QSPI
			1	Disables Data Input
SCK[1]	Master	DDQS2	0	Disables Clock Output
			1	Clock Output from QSPI
	Slave		0	Clock Input to QSPI
			1	Disables Clock Input
PCS0/$\overline{SS}$	Master	DDQS3	0	Assertion Causes Mode Fault
			1	Chip-Select Output
	Slave		0	QSPI Slave Select Input
			1	Disables Select Input
PCS[3:1]	Master	DDQS [4:6]	0	Disables Chip-Select Output
			1	Chip-Select Output
	Slave		0	Inactive
			1	Inactive
TXD[2]	Transmit	DDQS7	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 set), 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.

D.4.10 SPCR0 — QSPI Control Register 0 $YFFC18

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSTR	WOMQ	BITS				CPOL	CPHA	SP							
RESET:															
0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0

SPCR0 contains parameters for configuring the QSPI and enabling various modes of operation. The CPU has read/write access to SPCR0, but the QSM has read access only. SPCR0 must be initialized before QSPI operation begins. Writing a new value to SPCR0 while the QSPI is enabled disrupts operation.

MSTR — Master/Slave Mode Select
0 = QSPI is a slave device.
1 = QSPI is system master.

WOMQ — Wired-OR Mode for QSPI Pins
0 = Outputs have normal MOS drivers.
1 = Pins designated for output by DDRQS have open-drain drivers.

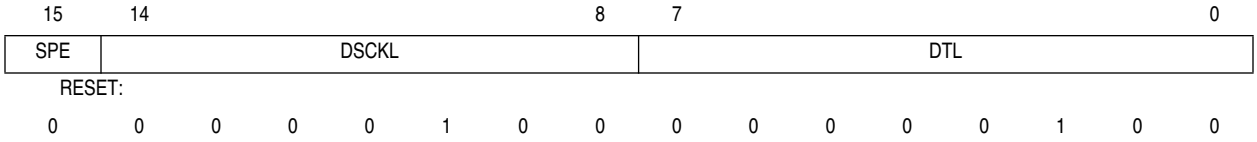
BITS — Bits Per Transfer
The BITS field determines the number of serial data bits transferred.

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.

CPHA — Clock Phase
0 = Data 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.

SPBR — Serial Clock Baud Rate
QSPI baud rate is selected by writing a value from 2 to 255 into SPBR. Giving BR a value of zero or one disables SCK (disable state determined by CPOL).

D.4.11 SPCR1 — QSPI Control Register 1 $YFFC1A



SPCR1 enables the QSPI and specified transfer delays. The CPU32 has read/write access to SPCR1, but the QSM has read access only to all bits but enable bit SPE. SPCR1 must be written last during initialization because it contains SPE. Writing a new value to SPCR1 while the QSPI is enabled disrupts operation.

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 — 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.

DTL — Length of Delay after Transfer
When the DT bit in command RAM is set, this field determines the length of delay after serial transfer.

D.4.12 SPCR2 — QSPI Control Register 2 $YFFC1C

15	14	13	12	11 … 8	7	6	5	4	3 … 0
SPIFIE	WREN	WRTO	0	ENDQP	0	0	0	0	NEWQP
RESET:									
0	0	0	0	0	0	0	0	0	0

SPCR2 contains QSPI queue pointers, wraparound mode control bits, and an interrupt enable bit. The CPU32 has read/write access to SPCR2, but the QSM has read access only. SPCR2 is buffered. New SPCR2 values become effective only after completion of the current serial transfer. Rewriting NEWQP in SPCR2 causes execution to restart at the designated location. SPCR2 reads return the value of the register, not the buffer.

SPIFIE — SPI Finished Interrupt Enable
- 0 = QSPI interrupts disabled
- 1 = QSPI interrupts enabled

WREN — Wrap Enable
- 0 = Wraparound mode disabled
- 1 = Wraparound mode enabled

WRTO — Wrap To
- 0 = Wrap to pointer address $0
- 1 = Wrap to address in NEWQP

ENDQP — Ending Queue Pointer
This field contains the last QSPI queue address.

NEWQP — New Queue Pointer Value
This field contains the first QSPI queue address.

D.4.13 SPCR3 — QSPI Control Register 3 $YFFC1E
SPSR — QSPI Status Register $YFFC1F

15	14	13	12	11	10	9	8	7	6	5	4	3 … 0
0	0	0	0	0	LOOPQ	HMIE	HALT	SPIF	MODF	HALTA	0	CPTQP
RESET:												
0	0	0	0	0	0	0	0	0	0	0	0	0

SPCR3 contains the loop mode enable bit, halt and mode fault interrupt enables, and the halt control bit. The CPU has read/write access to SPCR3, but the QSM has read access only. SPCR3 must be initialized before QSPI operation begins. Writing a new value to SPCR3 while the QSPI is enabled disrupts operation. SPSR contains information concerning the current serial transmission. Only the QSPI can set bits in SPSR. The CPU reads SPSR to obtain QSPI status information and writes it to clear status flags.

LOOPQ — QSPI Loop Mode
- 0 = Feedback path disabled
- 1 = Feedback path enabled

HMIE — HALTA and MODF Interrupt Enable
0 = HALTA and MODF interrupts disabled
1 = HALTA and MODF interrupts enabled

HALT — Halt
0 = Halt not enabled
1 = Halt enabled

SPIF — QSPI Finished Flag
0 = QSPI not finished
1 = QSPI finished

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).

HALTA — Halt Acknowledge Flag
0 = QSPI not halted
1 = QSPI halted

CPTQP — Completed Queue Pointer
CPTQP 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 contains either the reset value ($0) or a pointer to the last command completed in the previous queue.

D.4.14 RR[0:F] — Receive Data RAM $YFFD00–$YFFD0E

Data received by the QSPI is stored in this segment. The CPU32 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.

D.4.15 TR[0:F] — Transmit Data RAM $YFFD20–$YFFD3E

Data that is to be transmitted by the QSPI is stored in this segment. The CPU32 normally 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.

D.4.16 CR[0:F] — Command RAM $YFFD40–$YFFD4F

7	6	5	4	3	2	1	0
CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0*
—	—	—	—	—	—	—	—
CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0*
COMMAND CONTROL				PERIPHERAL CHIP SELECT			

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

Command RAM is used by the QSPI when in master mode. The CPU32 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 proceeds from the address in NEWQP through the address in ENDQP (both of these fields are in SPCR2).

PCS[3:0] — Peripheral Chip Select

Peripheral chip-select bits are used to select an external device for serial data transfer. More than one peripheral chip select may be activated at a time, and more than one peripheral chip can be connected to each PCS pin, provided proper fanout is observed. PCS0 shares a pin with the slave select ($\overline{SS}$) signal, which initiates slave mode serial transfer. If $\overline{SS}$ is taken low when the QSPI is in master mode, a mode fault occurs.

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 = Eight bits
1 = Number of bits set in BITS field of SPCR0

DT — Delay after Transfer

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

DSCK — PCS to SCK Delay

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

D.5 Time Processor Unit

Table D-5 is the TPU address map. The column labeled “Access” indicates the privilege level at which the CPU must be operating to access the register. A designation of “S” indicates that supervisor access is required: a designation of “S/U” indicates that the register can be programmed to the desired privilege level.

Table D-5 TPU Address Map

Access	Address	15 8	7 0
S	$YFFE00	TPU MODULE CONFIGURATION REGISTER (TPUMCR)	
S	$YFFE02	TEST CONFIGURATION REGISTER (TCR)	
S	$YFFE04	DEVELOPMENT SUPPORT CONTROL REGISTER (DSCR)	
S	$YFFE06	DEVELOPMENT SUPPORT STATUS REGISTER (DSSR)	
S	$YFFE08	TPU INTERRUPT CONFIGURATION REGISTER (TICR)	
S	$YFFE0A	CHANNEL INTERRUPT ENABLE REGISTER (CIER)	
S	$YFFE0C	CHANNEL FUNCTION SELECTION REGISTER 0 (CFSR0)	
S	$YFFE0E	CHANNEL FUNCTION SELECTION REGISTER 1 (CFSR1)	
S	$YFFE10	CHANNEL FUNCTION SELECTION REGISTER 2 (CFSR2)	
S	$YFFE12	CHANNEL FUNCTION SELECTION REGISTER 3 (CFSR3)	
S/U	$YFFE14	HOST SEQUENCE REGISTER 0 (HSQR0)	
S/U	$YFFE16	HOST SEQUENCE REGISTER 1 (HSQR1)	
S/U	$YFFE18	HOST SERVICE REQUEST REGISTER 0 (HSRR0)	
S/U	$YFFE1A	HOST SERVICE REQUEST REGISTER 1 (HSRR1)	
S	$YFFE1C	CHANNEL PRIORITY REGISTER 0 (CPR0)	
S	$YFFE1E	CHANNEL PRIORITY REGISTER 1 (CPR1)	
S	$YFFE20	CHANNEL INTERRUPT STATUS REGISTER (CISR)	
S	$YFFE22	LINK REGISTER (LR)	
S	$YFFE24	SERVICE GRANT LATCH REGISTER (SGLR)	
S	$YFFE26	DECODED CHANNEL NUMBER REGISTER (DCNR)	

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

D.5.1 TPUMCR — TPU Module Configuration Register $YFFE00

15	14	13	12	11	10	9	8	7	6	5	4	3				0
STOP	TCR1P		TCR2P		EMU	T2CG	STF	SUPV	PSCK	0	0	IARB				
RESET:																
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

STOP — Stop Bit

0 = TPU operating normally

1 = Internal clocks shut down

TCR1P — Timer Count Register 1 Prescaler Control

TCR1 is clocked from the output of a prescaler. The prescaler's input is the internal TPU system clock divided by either 4 or 32, depending on the value of the PSCK bit. The prescaler divides this input by 1, 2, 4, or 8. Channels using TCR1 have the capability to resolve down to the TPU system clock divided by 4.

TCR1 Prescaler	Divide By	PSCK = 0		PSCK = 1	
		Number of Clocks	Rate at 16 MHz	Number of Clocks	Rate at 16 MHz
00	1	32	2 ms	4	250 ns
01	2	64	4 ms	8	500 ns
10	4	128	8 ms	16	1 ms
11	8	256	16 ms	32	2 ms

TCR2P — Timer Count Register 2 Prescaler Control

TCR2 is clocked from the output of a prescaler. If T2CG = 0, the input to the TCR2 prescaler is the external TCR2 clock source. If T2CG = 1, the input is the TPU system clock divided by eight. The TCR2 field specifies the value of the prescaler: 1, 2, 4, or 8. Channels using TCR2 have the capability to resolve down to the TPU system clock divided by 8. The following table is a summary of prescaler output.

TCR2 Prescaler	Divide By	Internal Clock Divided By	External Clock Divided By
00	1	8	1
01	2	16	2
10	4	32	4
11	8	64	8

EMU — Emulation Control

In emulation mode, the TPU executes microinstructions from MCU TPURAM exclusively. Access to the TPURAM module through the IMB by a host is blocked, and the TPURAM module is dedicated for use by the TPU. After reset, this bit can be written only once.

0 = TPU and TPURAM not in emulation mode
1 = TPU and TPURAM in emulation mode

T2CG — TCR2 Clock/Gate Control

When the T2CG bit is set, the external TCR2 pin functions as a gate of the DIV8 clock (the TPU system clock divided by eight). In this case, when the external TCR2 pin is low, the DIV8 clock is blocked, preventing it from incrementing TCR2. When the external TCR2 pin is high, TCR2 is incremented at the frequency of the DIV8 clock. When T2CG is cleared, an external clock from the TCR2 pin, which has been synchronized and fed through a digital filter, increments TCR2.

0 = TCR2 pin used as clock source for TCR2
1 = TCR2 pin used as gate of DIV8 clock for TCR2

STF — Stop Flag

0 = TPU operating
1 = TPU stopped (STOP bit has been asserted)

SUPV — Supervisor/Unrestricted

0 = Supervisor access
1 = User access

PSCK — Prescaler Clock

0 = System clock/32 is input to TCR1 prescaler
1 = System clock/4 is input to TCR1 prescaler

IARB — Interrupt Arbitration

Each module that generates interrupts must have an IARB value. IARB values are used to arbitrate between interrupt requests of the same priority.

D.5.2 TCR — Test Configuration Register $YFFE02

The TCR is used for factory test of the MCU.

D.5.3 DSCR — Development Support Control Register $YFFE04

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HOT4	0	0	0	0	BLC	CLKS	FRZ1	FRZ0	CCL	BP	BC	BH	BL	BM	BT
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HOT4 — Hang on T4

0 = Exit wait on T4 state caused by assertion of HOT4
1 = Enter wait on T4 state

BLC — Branch Latch Control

0 = Latch conditions into branch condition register before exiting halted state.
1 = Do not latch conditions into branch condition register before exiting the halted state or during the time-slot transition period.

CLKS — Stop Clocks (to TCRs)

0 = Do not stop TCRs.
1 = Stop TCRs during the halted state.

FRZ[1:0] — IMB FREEZE Response

The FRZ bits specify the TPU microengine response to the FREEZE signal.

FRZ[1:0]	TPU Response
00	Ignore Freeze
01	Reserved
10	Freeze at End of Current Microcycle
11	Freeze at Next Time-Slot Boundary

CCL — Channel Conditions Latch

CCL controls the latching of channel conditions (MRL and TDL) when the CHAN register is written.

0 = Only the pin state condition of the new channel is latched as a result of the write CHAN register microinstruction.
1 = Pin state, MRL, and TDL conditions of the new channel are latched as a result of a write CHAN register microinstruction.

BP, BC, BH, BL, BM, and BT — Breakpoint Enable Bits

DSCR[5:0] are TPU breakpoint enables. Setting a bit enables a breakpoint condition.

BP — Break if mPC equals mPC breakpoint register.

BC — Break if CHAN register equals channel breakpoint register at beginning of state or when CHAN is changed through microcode.

BH — Break if host service latch is asserted at beginning of state.

BL — Break if link service latch is asserted at beginning of state.

BM — Break if MRL is asserted at beginning of state.

BT — Break if TDL is asserted at beginning of state.

D.5.4 DSSR — Development Support Status Register $YFFE06

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	BKPT	PCBK	CHBK	SRBK	TPUF	0	0	0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BKPT — Breakpoint Asserted Flag

If an internal breakpoint caused the TPU to enter the halted state, the TPU asserts the BKPT signal on the IMB and the BKPT flag. The TPU continues to assert BKPT until it recognizes a breakpoint acknowledge cycle from a host, or until the FREEZE signal on the IMB is asserted.

PCBK — µPC Breakpoint Flag

PCBK is asserted if a breakpoint occurs because of a µPC register match with the µPC breakpoint register. PCBK is negated when the BKPT flag is negated.

CHBK — Channel Register Breakpoint Flag

CHBK is asserted if a breakpoint occurs because of a CHAN register match with the channel register breakpoint register. CHBK is negated when the BKPT flag is negated.

SRBK — Service Request Breakpoint Flag

SRBK is asserted if a breakpoint occurs because of any of the service request latches being asserted along with their corresponding enable flag in the development support control register. SRBK is negated when the BKPT flag is negated.

TPUF — TPU FREEZE Flag

TPUF is asserted whenever the TPU is in a halted state as a result of FREEZE being asserted. This flag is automatically negated when the TPU exits the halted state because of FREEZE being negated.

D.5.5 TICR — TPU Interrupt Configuration Register $YFFE08

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

CIRL — Channel Interrupt Request Level

This three-bit encoded field specifies the interrupt request level for all channels. Level seven for this field indicates a nonmaskable interrupt; level zero indicates that all channel interrupts are disabled.

CIBV — Channel Interrupt Base Vector

The TPU is assigned 16 unique interrupt vector numbers, one vector number for each channel. The CIBV field specifies the most significant nibble of all 16 TPU channel interrupt vector numbers. The lower nibble of the TPU interrupt vector number is determined by the channel number on which the interrupt occurs.

D.5.6 CIER — Channel Interrupt Enable Register $YFFE0A

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

CH[15:0] — Channel Interrupt Enable/Disable

0 = Channel interrupts disabled
1 = Channel interrupts enabled

D.5.7 CFSR0 — Channel Function Select Register 0 $YFFE0C

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHANNEL15				CHANNEL14				CHANNEL13				CHANNEL12			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

D.5.8 CFSR1 — Channel Function Select Register 1 $YFFE0E

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHANNEL11				CHANNEL10				CHANNEL9				CHANNEL8			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

D.5.9 CFSR2 — Channel Function Select Register 2 $YFFE10

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHANNEL7				CHANNEL6				CHANNEL5				CHANNEL4			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

D.5.10 CFSR3 — Channel Function Select Register 3 $YFFE12

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHANNEL3				CHANNEL2				CHANNEL1				CHANNEL0			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CHANNEL[15:0] — Encoded Time Function for each Channel

Encoded four-bit fields in the channel function select registers specify one of 16 time functions to be executed on the corresponding channel.

D.5.11 HSQR0 — Host Sequence Register 0 $YFFE14

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

D.5.12 HSQR1 — Host Sequence Register 1 $YFFE16

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

CH[15:0] — Encoded Host Sequence

The host sequence field selects the mode of operation for the time function selected on a given channel. The meaning of the host sequence bits depends on the time function specified.

D.5.13 HSRR0 — Host Service Request Register 0 $YFFE18

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

D.5.14 HSRR1 — Host Service Request Register 1 $YFFE1A

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

CH[15:0] — Encoded Type of Host Service

The host service request field selects the type of host service request for the time function selected on a given channel. The meaning of the host service request bits depends on the time function specified.

A host service request field cleared to %00 signals the host that service is completed by the microengine on that channel. The host can request service on a channel by writing the corresponding host service request field to one of three nonzero states. The CPU should monitor the host service request register until the TPU clears the service request to %00 before the CPU changes any parameters or issues a new service request to the channel.

D.5.15 CPR0 — Channel Priority Register 0 $YFFE1C

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH 15		CH 14		CH13		CH 12		CH 11		CH 10		CH 9		CH 8	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

D.5.16 CPR1 — Channel Priority Register 1 $YFFE1E

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

CH[15:0] — Encoded One of Three Channel Priority Levels

CHX[1:0]	Service	Guaranteed Time Slots
00	Disabled	—
01	Low	1 out of 7
10	Middle	2 out of 7
11	High	4 out of 7

D.5.17 CISR — Channel Interrupt Status Register $YFFE20

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

CH[15:0] — Channel Interrupt Status Bit
0 = Channel interrupt not asserted
1 = Channel interrupt asserted

D.5.18 LR — Link Register $YFFE22

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

CH[15:0] — Test Mode Link Service Request Enable Bit
0 = Link bit not asserted
1 = Link bit asserted

D.5.19 SGLR — Service Grant Latch Register $YFFE24

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

CH[15:0] — Service Granted Bits

D.5.20 DCNR — Decoded Channel Number Register $YFFE26

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

CH[15:0] — Service Status Bits

D.5.21 TPU Parameter RAM

The channel parameter registers are organized as one hundred 16-bit words of RAM. Channels 0 to 13 have six parameters. Channels 14 and 15 each have eight parameters. The parameter registers constitute a shared work space for communication between the bus master and the TPU.

Table D-6 Parameter RAM Address Map

Channel Number	Base Address	Parameter							
		0	1	2	3	4	5	6	7
0	$YFFF##	00	02	04	06	08	0A	—	—
1	$YFFF##	10	12	14	16	18	1A	—	—
2	$YFFF##	20	22	24	26	28	2A	—	—
3	$YFFF##	30	32	34	36	38	3A	—	—
4	$YFFF##	40	42	44	46	48	4A	—	—
5	$YFFF##	50	52	54	56	58	5A	—	—
6	$YFFF##	60	62	64	66	68	6A	—	—
7	$YFFF##	70	72	74	76	78	7A	—	—
8	$YFFF##	80	82	84	86	88	8A	—	—
9	$YFFF##	90	92	94	96	98	9A	—	—
10	$YFFF##	A0	A2	A4	A6	A8	AA	—	—
11	$YFFF##	B0	B2	B4	B6	B8	BA	—	—
12	$YFFF##	C0	C2	C4	C6	C8	CA	—	—
13	$YFFF##	D0	D2	D4	D6	D8	DA	—	—
14	$YFFF##	E0	E2	E4	E6	E8	EA	EC	EE
15	$YFFF##	F0	F2	F4	F6	F8	FA	FC	FE

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

Table D-7 MC68332 Module Address Map

(Assumes SIMCR MM = 1)

SIM			
Access	**Address**	**15 8**	**7 0**
S	$FFFA00	MODULE CONFIGURATION (SIMCR)	
S	$FFFA02	FACTORY TEST (SIMTR)	
S	$FFFA04	CLOCK SYNTHESIZER CONTROL (SYNCR)	
S	$FFFA06	NOT USED	RESET STATUS (RSR)
S	$FFFA08	MODULE TEST E (SIMTRE)	
S	$FFFA0A	NOT USED	NOT USED
S	$FFFA0C	NOT USED	NOT USED
S	$FFFA0E	NOT USED	NOT USED
S/U	$FFFA10	NOT USED	PORTE DATA (PORTE0)
S/U	$FFFA12	NOT USED	PORTE DATA (PORTE1)
S/U	$FFFA14	NOT USED	PORTE DATA DIRECTION (DDRE)
S	$FFFA16	NOT USED	PORTE PIN ASSIGNMENT (PEPAR)
S/U	$FFFA18	NOT USED	PORTF DATA (PORTF0)
S/U	$FFFA1A	NOT USED	PORTF DATA (PORTF1)
S/U	$FFFA1C	NOT USED	PORTF DATA DIRECTION (DDRF)
S	$FFFA1E	NOT USED	PORTF PIN ASSIGNMENT (PFPAR)
S	$FFFA20	NOT USED	SYSTEM PROTECTION CONTROL (SYPCR)
S	$FFFA22	PERIODIC INTERRUPT CONTROL (PICR)	
S	$FFFA24	PERIODIC INTERRUPT TIMING (PITR)	
S	$FFFA26	NOT USED	SOFTWARE SERVICE (SWSR)
S	$FFFA28	NOT USED	NOT USED
S	$FFFA2A	NOT USED	NOT USED
S	$FFFA2C	NOT USED	NOT USED
S	$FFFA2E	NOT USED	NOT USED
S	$FFFA30	TEST MODULE MASTER SHIFT A (TSTMSRA)	
S	$FFFA32	TEST MODULE MASTER SHIFT B (TSTMSRB)	
S	$FFFA34	TEST MODULE SHIFT COUNT (TSTSC)	
S	$FFFA36	TEST MODULE REPETITION COUNTER (TSTRC)	
S	$FFFA38	TEST MODULE CONTROL (CREG)	
S/U	$FFFA3A	TEST MODULE DISTRIBUTED (DREG)	
	$FFFA3C	NOT USED	NOT USED
	$FFFA3E	NOT USED	NOT USED
S/U	$FFFA40	NOT USED	PORT C DATA (PORTC)
	$FFFA42	NOT USED	NOT USED
S	$FFFA44	CHIP-SELECT PIN ASSIGNMENT (CSPAR0)	
S	$FFFA46	CHIP-SELECT PIN ASSIGNMENT (CSPAR1)	
S	$FFFA48	CHIP-SELECT BASE BOOT (CSBARBT)	
S	$FFFA4A	CHIP-SELECT OPTION BOOT (CSORBT)	
S	$FFFA4C	CHIP-SELECT BASE 0 (CSBAR0)	
S	$FFFA4E	CHIP-SELECT OPTION 0 (CSOR0)	
S	$FFFA50	CHIP-SELECT BASE 1 (CSBAR1)	
S	$FFFA52	CHIP-SELECT OPTION 1 (CSOR1)	
S	$FFFA54	CHIP-SELECT BASE 2 (CSBAR2)	
S	$FFFA56	CHIP-SELECT OPTION 2 (CSOR2)	
	$FFFA58	CHIP-SELECT BASE 3 (CSBAR3)	

Table D-7 MC68332 Module Address Map (Continued)

(Assumes SIMCR MM = 1)

SIM			
Access	**Address**	**15 8**	**7 0**
S	$FFFA5A	CHIP-SELECT OPTION 3 (CSOR3)	
S	$FFFA5C	CHIP-SELECT BASE 4 (CSBAR4)	
S	$FFFA5E	CHIP-SELECT OPTION 4 (CSOR4)	
S	$FFFA60	CHIP-SELECT BASE 5 (CSBAR5)	
S	$FFFA62	CHIP-SELECT OPTION 5 (CSOR5)	
S	$FFFA64	CHIP-SELECT BASE 6 (CSBAR6)	
S	$FFFA66	CHIP-SELECT OPTION 6 (CSOR6)	
S	$FFFA68	CHIP-SELECT BASE 7 (CSBAR7)	
S	$FFFA6A	CHIP-SELECT OPTION 7 (CSOR7)	
S	$FFFA6C	CHIP-SELECT BASE 8 (CSBAR8)	
S	$FFFA6E	CHIP-SELECT OPTION 8 (CSOR8)	
S	$FFFA70	CHIP-SELECT BASE 9 (CSBAR9)	
S	$FFFA72	CHIP-SELECT OPTION 9 (CSOR9)	
S	$FFFA74	CHIP-SELECT BASE 10 (CSBAR10)	
S	$FFFA76	CHIP-SELECT OPTION 10 (CSOR10)	
	$FFFA78	NOT USED	NOT USED
	$FFFA7A	NOT USED	NOT USED
	$FFFA7C	NOT USED	NOT USED
	$FFFA7E	NOT USED	NOT USED
TPURAM			
Access	**Address**	**15 8**	**7 0**
S	$FFFB00	TPURAM MODULE CONFIGURATION REGISTER (TRAMMCR)	
S	$FFFB02	TPURAM TEST REGISTER (TRAMTST)	
S	$FFFB04	TPURAM BASE ADDRESS AND STATUS REGISTER (TRAMBAR)	
S	$FFFB06	NOT USED	
QSM			
Access	**Address**	**15 8**	**7 0**
S	$FFFC00	QSM MODULE CONFIGURATION (QSMCR)	
S	$FFFC02	QSM TEST (QTEST)	
S	$FFFC04	QSM INTERRUPT LEVEL (QILR)	QSM INTERRUPT VECTOR (QIVR)
S/U	$FFFC06	NOT USED	
S/U	$FFFC08	SCI CONTROL 0 (SCCR0)	
S/U	$FFFC0A	SCI CONTROL 1 (SCCR1)	
S/U	$FFFC0C	SCI STATUS (SCSR)	
S/U	$FFFC0E	SCI DATA (SCDR)	
S/U	$FFFC10	NOT USED	
S/U	$FFFC12	NOT USED	
S/U	$FFFC14	NOT USED	PQS DATA (PORTQS)
S/U	$FFFC16	PQS PIN ASSIGNMENT (PQSPAR)	PQS DATA DIRECTION (DDRQS)
S/U	$FFFC18	SPI CONTROL 0 (SPCR0)	
S/U	$FFFC1A	SPI CONTROL 1 (SPCR1)	
S/U	$FFFC1C	SPI CONTROL 2 (SPCR2)	
S/U	$FFFC1E	SPI CONTROL 3 (SPCR3)	SPI STATUS (SPSR)
S/U	$FFFC20–$FFFCFF	NOT USED	
S/U QUEUE RAM	$FFFD00–$FFFD1F	RECEIVE RAM (RR[0:F])	

Table D-7 MC68332 Module Address Map (Continued)

(Assumes SIMCR MM = 1)

QSM			
Access	**Address**	**15 8**	**7 0**
S/U QUEUE RAM	$FFFD20–$FFFD3F	TRANSMIT RAM (TR[0:F])	
S/U QUEUE RAM	$FFFD40–$FFFD4F	COMMAND RAM (CR[0:F])	
TPU			
Access	**Address**	**15 8**	**7 0**
S	$FFFE00	TPU MODULE CONFIGURATION REGISTER (TPUMCR)	
S	$FFFE02	TEST CONFIGURATION REGISTER (TCR)	
S	$FFFE04	DEVELOPMENT SUPPORT CONTROL REGISTER (DSCR)	
S	$FFFE06	DEVELOPMENT SUPPORT STATUS REGISTER (DSSR)	
S	$FFFE08	TPU INTERRUPT CONFIGURATION REGISTER (TICR)	
S	$FFFE0A	CHANNEL INTERRUPT ENABLE REGISTER (CIER)	
S	$FFFE0C	CHANNEL FUNCTION SELECTION REGISTER 0 (CFSR0)	
S	$FFFE0E	CHANNEL FUNCTION SELECTION REGISTER 1 (CFSR1)	
S	$FFFE10	CHANNEL FUNCTION SELECTION REGISTER 2 (CFSR2)	
S	$FFFE12	CHANNEL FUNCTION SELECTION REGISTER 3 (CFSR3)	
S/U	$FFFE14	HOST SEQUENCE REGISTER 0 (HSQR0)	
S/U	$FFFE16	HOST SEQUENCE REGISTER 1 (HSQR1)	
S/U	$FFFE18	HOST SERVICE REQUEST REGISTER 0 (HSRR0)	
S/U	$FFFE1A	HOST SERVICE REQUEST REGISTER 1 (HSRR1)	
S	$FFFE1C	CHANNEL PRIORITY REGISTER 0 (CPR0)	
S	$FFFE1E	CHANNEL PRIORITY REGISTER 1 (CPR1)	
S	$FFFE20	CHANNEL INTERRUPT STATUS REGISTER (CISR)	
S	$FFFE22	LINK REGISTER (LR)	
S	$FFFE24	SERVICE GRANT LATCH REGISTER (SGLR)	
S	$FFFE26	DECODED CHANNEL NUMBER REGISTER (DCNR)	

TPU Parameter RAM									
Channel Number	**Base Address**	**Parameter**							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	$FFFFF##	00	02	04	06	08	0A	—	—
1	$FFFFF##	10	12	14	16	18	1A	—	—
2	$FFFFF##	20	22	24	26	28	2A	—	—
3	$FFFFF##	30	32	34	36	38	3A	—	—
4	$FFFFF##	40	42	44	46	48	4A	—	—
5	$FFFFF##	50	52	54	56	58	5A	—	—
6	$FFFFF##	60	62	64	66	68	6A	—	—
7	$FFFFF##	70	72	74	76	78	7A	—	—
8	$FFFFF##	80	82	84	86	88	8A	—	—
9	$FFFFF##	90	92	94	96	98	9A	—	—
10	$FFFFF##	A0	A2	A4	A6	A8	AA	—	—
11	$FFFFF##	B0	B2	B4	B6	B8	BA	—	—
12	$FFFFF##	C0	C2	C4	C6	C8	CA	—	—
13	$FFFFF##	D0	D2	D4	D6	D8	DA	—	—
14	$FFFFF##	E0	E2	E4	E6	E8	EA	EC	EE
15	$FFFFF##	F0	F2	F4	F6	F8	FA	FC	FE

Table D-8 Register Bit and Field Mnemonics

Mnemonic	Name	Register Location
ADDR[23:11]	Base Address	CSBAR[0:10], CSBARBT, TRAMBAR
$\overline{\text{AVEC}}$	Autovector Enable	CSOR[0:10], CSORBT
BP, BC, BH, BL, BM, BT	Breakpoint Enable Points	DSCR
BITS	Bits Per Transfer	SPCR0
BITSE	Bits Per Transfer Enable	CR[0:F]
BKPT	Breakpoint Asserted Flag	DSSR
BLC	Branch Latch Control	DSCR
BLKSZ	Block Size	CSBAR[0:10], CSBARBT
BME	Bus Monitor External Enable	SYPCR
BMT[1:0]	Bus Monitor Timing	SYPCR
BYTE	Upper/Lower Byte Option	CSOR[0:10], CSORBT
C	Carry Flag	CCR
CCL	Channel Conditions Latch	DSCR
CIBV	Channel Interrupt Base Vector	TICR
CIRL	Channel Interrupt Request Level	TICR
CH[15:0]	Channel Function Select	CFSR[0:3]
CH[15:0]	Channel Interrupt Enable/Disable	CIER
CH[15:0]	Channel Interrupt Status	CISR
CH[15:0]	Channel Priority	CPR[0:1]
CH[15:0]	Service Status	DCNR
CH[15:0]	Encoded Host Sequence	HSQR[0:1]
CH[15:0]	Host Service Request	HSRR[0:1]
CH[15:0]	Link	LR
CH[15:0]	Service Granted	SGLR
CHBK	Channel Register Breakpoint Flag	DSSR
CLKS	Stop Clocks	DSCR
CONT	Continue	CR[0:F]
CPHA	Clock Phase	SPCR0
CPOL	Clock Polarity	SPCR0
CPTQP	Completed Queue Pointer	SPSR
CSPA0[6:1]	Chip-Select [6:1]	CSPAR0
CSPA1[4:0]	Chip-Select [4:0]	CSPAR1
$\overline{\text{CSBOOT}}$	Boot ROM Chip Select	CSPAR0
DDE[7:0]	Port E Data Direction	DDRE
DDF[7:0]	Port F Data Direction	DDRF
DDQS[7:0]	Port QS Data Direction	DDRQS
$\overline{\text{DSACK}}$	Data Strobe Acknowledge	CSOR[0:10], CSORBT
DSCK	PCS to SCK Delay	CR[0:F]
DSCKL	Delay Before SCK	SPCR1
DT	Delay After Transfer	CR[0:F]
DTL	Length of Delay After Transfer	SPCR1
EMU	Emulation Control	TPUMCR
EDIV	ECLK Divide Rate	SYNCR
ENDQP	Ending Queue Pointer	SPCR2
EXOFF	External Clock Off	SIMCR
EXT	External Reset	RSR

Table D-8 Register Bit and Field Mnemonics (Continued)

Mnemonic	Name	Register Location
FE	Framing Error	SCSR
FRZBM	Freeze Bus Monitor Enable	SIMCR
FRZSW	Freeze Software Enable	SIMCR
FRZ[1:0]	Freeze Control	DSCR, QSMCR
HALT	Halt	SPCR3
HALTA	Halt Acknowledge Flag	SPSR
HLT	Halt Monitor Reset	RSR
HME	Halt Monitor Enable	SYPCR
HMIE	HALTA and MODF Interrupt Enable	SPCR3
HOT4	Hang on T4	DSCR
IARB[3:0]	Interrupt Arbitration Field	QSMCR, SIMCR, TPUMCR
IDLE	Idle-Line Detected	SCSR
ILIE	Idle-Line Interrupt Enable	SCCR1
ILQSPI	Interrupt Level for QSPI	QILR
ILSCI	Interrupt Level for SCI	QILR
ILT	Idle-Line Detect Type	SCCR1
INTV[7:0]	Interrupt Vector Number	QIVR
IP[2:0]	Interrupt Priority Mask	SR
IPL	Interrupt Priority Level	CSOR[0:10], CSORBT
LOC	Loss of Clock Reset	RSR
LOOPQ	QSPI Loop Mode	SPCR3
LOOPS	Loop Mode	SCCR1
M	Mode Select	SCCR1
MM	Module Mapping	SIMCR
MODE	Asynchronous/Synchronous Mode	CSOR[0:10], CSORBT
MODF	Mode Fault Flag	SPSR
MSTR	Master/Slave Mode Select	SPCR0
N	Negative Flag	CCR
NEWQP	New Queue Pointer Value	SPCR2
NF	Noise Error	SCSR
OR	Overrun Error	SCSR
PC[6:0]	Port C Data	PORTC
PCBK	µPC Breakpoint Flag	DSSR
PCS[3:0]	Peripheral Chip Select	CR[0:F]
PE	Parity Enable	SCCR1
PE[7:0]	Port E Data	PORTE
PEPA[7:0]	Port E Pin Assignment	PEPAR
PF	Parity Error	SCSR
PF[7:0]	Port F Data	PORTF
PFPA[7:0]	Port F Pin Assignment	PFPAR
PIRQL[2:0]	Periodic Interrupt Request Level	PICR
PITM[7:0]	Periodic Interrupt Timing Modulus	PITR
PIV[7:0]	Periodic Interrupt Vector	PICR
POW	Power-Up Reset	RSR
PQS[7:0]	Port QS Data	PORTQS
PQSPA[6:0]	Port QS Pin Assignment	PQSPAR
PSCK	Prescaler Clock	TPUMCR
PT	Parity Type	SCCR1

Table D-8 Register Bit and Field Mnemonics (Continued)

Mnemonic	Name	Register Location
PTP	Periodic Timer Prescaler Control	PITR
RAF	Receiver Active	SCSR
RAMDS	TPURAM Array Disable	TRAMBAR
RASP[1:0]	TPURAM Array Space	TRAMMCR
RDRF	Receive Data Register Full	SCSR
RE	Receiver Enable	SCCR1
RIE	Receiver Interrupt Enable	SCCR1
RR[0:F]	Receive Data RAM	QSPI RAM
RSTEN	Reset Enable	SYNCR
R/$\overline{W}$	Read/Write	CSOR[0:10], CSORBT
RWU	Receiver Wakeup	SCCR1
R[8:0]/T[8:0]	SCI Receive/Transmit Data	SCDR
S	Supervisor/User State	SR
SBK	Send Break	SCCR1
SCBR	SCI Baud Rate	SCCR0
SHEN[1:0]	Show Cycle Enable	SIMCR
SLIMP	LIMP Mode	SYNCR
SLOCK	Synthesizer Lock	SYNCR
SLVEN	Factory Test Mode Enabled	SIMCR
SPACE	Address Space Select	CSOR[0:10], CSORBT
SPBR	Serial Clock Baud Rate	SPCR0
SPE	QSPI Enable	SPCR1
SPIF	QSPI Finished Flag	SPSR
SPIFIE	SPI Finished Interrupt Enable	SPCR2
SRBK	Service Request Breakpoint Flag	DSSR
STEXT	Stop Mode External Clock	SYNCR
STF	Stop Flag	TPUMCR
STOP	Stop Enable	QSMCR, TPUMCR, TRAMMCR
STRB	Address Strobe/Data Strobe	CSOR[0:10], CSORBT
STSIM	Stop Mode System Integration Clock	SYNCR
SUPV	Supervisor/Unrestricted	QSMCR, SIMCR, TPUMCR
SW	Software Watchdog Reset	RSR
SWE	Software Watchdog Enable	SYPCR
SWP	Software Watchdog Prescale	SYPCR
SWT[1:0]	Software Watchdog Timing	SYPCR
SYS	System Reset	RSR
T[1:0]	Trace Enable	SR
T2CG	TCR2 Clock/Gate Control	TPUMCR
TC	Transmit Complete	SCSR
TCIE	Transmit Complete Interrupt Enable	SCCR1
TCR1P	TCR1 Prescaler Control	TPUMCR
TCR2P	TCR2 Prescaler Control	TPUMCR
TDRE	Transmit Data Register Empty	SCSR
TE	Transmitter Enable	SCCR1
TIE	Transmit Interrupt Enable	SCCR1
TPUF	TPU FREEZE Flag	DSSR
TR[0:F]	Transmit Data RAM	QSPI RAM
TST	Test Submodule Reset	RSR

Table D-8 Register Bit and Field Mnemonics (Continued)

Mnemonic	Name	Register Location
V	Overflow Flag	CCR
W	Frequency Control (VCO)	SYNCR
WAKE	Wakeup by Address Mark	SCCR1
WOMQ	Wired-OR Mode for QSPI Pins	SPCR0
WOMS	Wired-OR Mode for SCI Pins	SCCR1
WREN	Wrap Enable	SPCR2
WRTO	Wrap To	SPCR2
X	Extend	CCR
X	Frequency Control Bit (Prescale)	SYNCR
Y[5:0]	Frequency Control (Counter)	SYNCR
Z	Zero Flag	CCR

SUMMARY OF CHANGES

This is a complete revision, with complete reprint. All known errors in the publication have been corrected. The following summary lists significant changes.

Section 1 Introduction

Page 1-1	New introduction to the MC68332 microcontroller.

Section 2 Nomenclature

Page 2-1	Added “Symbols and Operators” section.
Page 2-2	Added “CPU32 Registers” section.
Pages 2-3 – 2-4	Added “Pin and Signal Mnemonics” section.
Pages 2-5 – 2-6	Added “Register Mnemonics” section.
Page 2-7	Added “Conventions” section.

Section 3 Overview

Page 3-3	New block diagram drawn.
Pages 3-4 – 3-5	New 132-pin package and 144-pin package pin assignment diagrams drawn.
Pages 3-6 – 3-7	Revised pin characteristics.
Page 3-8	Incorporated V_{DD}/V_{SS} breakout information.
Pages 3-9 – 3-11	Revised signal characteristics and signal function tables.
Pages 3-12 – 3-17	New system memory maps drawn.

Section 4 System Integration Module

Pages 4-1 – 4-70	Expanded and revised SIM section. Made all register diagrams and bit mnemonics consistent. Incorporated new information concerning the system clock, resets, interrupts, and chip-select circuits.

Section 5 Central Processing Unit

Pages 5-1 – 5-30	Expanded and revised CPU section. Made all register diagrams and bit mnemonics consistent. Revised instruction set summary information.

Section 6 Queued Serial Module

Pages 6-1 – 6-36	Expanded and revised QSM section. Made all register diagrams and bit mnemonics consistent. Added information concerning SPI and SCI operation.

Section 7 Time Processor Unit

Pages 7-1 – 7-18	Expanded and revised TPU section. Made all register diagrams and bit mnemonics consistent. Revised time functions information to include both MC68332A and MC68332G microcode ROM applications.

Section 8 Standby RAM with TPU Emulation

Pages 8-1 – 8-4	Revised Standby RAM with TPU Emulation section. Made all register diagrams and bit mnemonics consistent.

Appendix a Electrical Characteristics

Pages A-1 – A-30	Completely revised electrical characteristics section. Added 20.97 MHz timing parameters.

Appendix B Mechanical Data and Ordering Information

Pages B-1 – B-10	Revised MC68332 132-pin assignment drawing. Included new diagrams on 144-pin assignment and package dimensions. Revised ordering information table.

Appendix C Development Support

Pages C-1 – C-2	New information on the M68MMDS1632 Modular Development System and the M68MEVB1632 Modular Evaluation Board.

Appendix D Register Summary

Pages D-1 – D-50	Revised address maps and register diagrams. Includes a new register bit and field mnemonics table.

INDEX

–F–

–G–

–H–

–I–

–L–

–M–

–S–

–T–

–U–

–V–

–W–

–X–

–Z–