MC68HC16Z1UM/AD

M68HC16 Family

MC68HC16Z1

USER'S MANUAL

MOTOROLA

MC68HC16Z1 USER'S MANUAL MOTOROLA

MC68HC16Z1
USER'S MANUAL

TABLE OF CONTENTS

Paragraph **Title** **Page**

Section 1
Introduction

Section 2
Nomenclature

Section 3
Overview

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

Section 4
System Integration Module

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

Section 5
Central Processing Unit

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

Section 6
Analog-to-Digital Converter

Paragraph Title Page

Section 7
Queued Serial Module

TABLE OF CONTENTS
(Continued)

Section 8
General-Purpose Timer

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

Section 9
Standby RAM Module

Appendix A
Electrical Characteristics

Appendix B
Mechanical Data and Ordering Information

Appendix C
Development Support

Appendix D
Register Summary

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS
(Continued)

LIST OF ILLUSTRATIONS
(Continued)

LIST OF TABLES

LIST OF TABLES
(Continued)

LIST OF TABLES
(Continued)

SECTION 1
INTRODUCTION

The MC68HC16Z1 is a high-speed 16-bit control unit that is upwardly code compatible with M68HC11 controllers. It is a member of the M68HC16 Family of modular microcontrollers.

M68HC16 controllers are built up from standard modules that interface via a common internal bus. Standardization facilitates rapid development of devices tailored for specific applications.

The MC68HC16Z1 incorporates a true 16-bit central processing unit (CPU16), a system integration module (SIM), an 8/10-bit analog-to-digital converter (ADC), a queued serial module (QSM), a general-purpose timer (GPT), and a 1024-byte standby RAM (SRAM). These modules are interconnected by the intermodule bus (IMB).

Maximum system clock for the MC68HC16Z1 is 16.78 MHz. A phase-locked loop circuit synthesizes the clock from a frequency reference. Either a crystal (nominal frequency: 32.768 kHz) or an externally generated signal can be used. System hardware and software support changes in clock rate during operation. Because the MC68HC16Z1 is a fully static design, 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 MC68HC16Z1 low. Power consumption can be minimized by stopping the system clock. The M68HC16 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 device has a comprehensive user's manual which provides sufficient information for normal operation of the device. The user's manual is supplemented by module reference manuals that provide detailed information about module operation and applications. Refer to Motorola publication *Advanced Microcontroller Unit (AMCU) Literature* (BR1116/D) for a complete listing of documentation to supplement this manual.

SECTION 2
NOMENCLATURE

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

2.1 Symbols and Operators

+ — Addition
– — Subtraction or negation (twos 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 CPU16 Registers

A — Accumulator A
AM — Accumulator M
B — Accumulator B
CCR — Condition code register
D — Accumulator D
E — Accumulator E
EK — Extended addressing extension field
IR — MAC multiplicand register
HR — MAC multiplier register
IX — Index register X
IY — Index register Y
IZ — Index register Z
K — Address extension register
PC — Program counter
PK — Program counter extension field
SK — Stack pointer extension field
SL — Multiply and accumulate sign latch
SP — Stack pointer
XK — Index register X extension field
YK — Index register Y extension field
ZK — Index register Z extension field
XMSK — Modulo addressing index register X mask
YMSK — Modulo addressing index register Y mask
S — Stop disable control bit
MV — AM overflow indicator
H — Half carry indicator
EV — AM extended overflow indicator
N — Negative indicator
Z — Zero indicator
V — Twos complement overflow indicator
C — Carry/borrow indicator
IP — Interrupt priority field
SM — Saturation mode control bit
PK — Program counter extension field

2.3 Pin and Signal Mnemonics

AN[7:0] — ADC Analog Inputs
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
ECLK — 6800 Bus Clock
DSI — Development Serial Input
DSO — Development Serial Output
EXTAL — External Crystal Oscillator Connection
FC[2:0] — Function Codes
FREEZE — Freeze
$\overline{HALT}$ — Halt
IC[4:1] — Input Capture
IPIPE0/IPIPE1 — Instruction Pipeline MUX
$\overline{IRQ[7:0]}$ — Interrupt Request
MISO — Master In Slave Out
MODCLK — Clock Mode Select
MOSI — Master Out Slave In
OC[5:1] — Output Compare
PADA[7:0] — ADC I/O Port A
PAI — Pulse Accumulator Input
PCLK — Pulse Accumulator Clock
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

PGP[7:0] — GPT I/O Port
PQS[7:0] — QSM I/O Port
PWMA, PWMB — Pulse Width Modulator Output
QUOT — Quotient Out
R/$\overline{W}$ — Read/Write
$\overline{RESET}$ — Reset
RXD — SCI Receive Data
SCK — QSPI Serial Clock
SIZ[1:0] — Size
$\overline{SS}$ — Slave Select
TSC — Three-State Control
$\overline{TSTME}$ — Test Mode Enable
TXD — SCI Transmit Data
V_{DDA}/V_{SSA} — A/D Converter Power
V_{DDSYN} — Clock Synthesizer Power
V_{RH}/V_{RL} — A/D Reference Voltage
V_{SSE}/V_{DDE} — External Peripheral Power (Source and Drain)
V_{SSI}/V_{DDI} — Internal Module Power (Source and Drain)
V_{STBY} — Standby RAM Power
XFC — External Filter Capacitor Connection
XTAL — External Crystal Oscillator Connection

2.4 Register Mnemonics

ADCMCR — ADC Module Configuration Register
ADCTL0 — A/D Control Register 0
ADCTL1 — A/D Control Register 1
ADSTAT — ADC Status Register
ADTEST — ADC Test Register
CFORC — Compare Force Register
CREG — Test Submodule Control Register
CSBARBT — Chip Select Base Address Register Boot ROM
CSBAR[0:10] — Chip Select Base Address Registers
CSORBT — Chip Select Option Register Boot ROM
CSPAR0 — Chip Select Pin Assignment Register 0
CSPAR1 — Chip Select Pin Assignment Register 1
DDRE — Port E Data Direction Register
DDRF — Port F Data Direction Register
DDRGP — Port GP Data Direction Register
PORTQS — Data Direction Register
DREG — Distributed Register
GPTMCR — GPT Module Configuration Register
GPTMTR — GPT Module Test Register (Reserved)
ICR — GPT Interrupt Configuration Register
OC1D — OC1 Action Data Register
OC1M — OC1 Action Mask Register
PACNT — Pulse Accumulator Counter
PACNT — Pulse Accumulator Counter
PACTL — Pulse Accumulator Control Register
PEPAR — Port E Pin Assignment Register
PFPAR — Port F Pin Assignment Register
PICR — Periodic Interrupt Control Register
PITR — Periodic Interrupt Timer Register
PORTADA — Port ADA Data Register
PORTC — Port C Data Register
PORTE — Port E Data Register
PORTF — Port F Data Register
PORTGP — Port GP Data Register
PORTQS — Port QS Data Register
PQSPAR — Port QS Pin Assignment Register
PRESCL — GPT Prescaler
PWMA — PWM Control Register A
PWMB — PWM Control Register B
PWMBUFA — PWM Buffer Register A
PWMBUFB — PWM Buffer Register B
PWMC — PWM Control Register C
PWMCNT — PWM Count Register

QILR — QSM Interrupt Level Register
QIVR — QSM Interrupt Vector Register
QSMCR — QSM Configuration Register
QTEST — QSM Test Register
RAMBAH — Array Base Address Register High
RAMBAL — Array Base Address Register Low
RAMMCR — RAM Module Configuration Register
RAMTST — RAM Test Register
RSLT[0:7] — ADC Result Registers
RSR — Reset Status Register
SCCR0 — SCI Control Register 0
SCCR1 — SCI Control Register 1
SCDR — SCI Data Register
SCSR — SCI Status Register
SIMCR — Module Configuration Register
SIMTR — System Integration Test Register
SIMTRE — System Integration Test Register (ECLK)
SPCR0 — QSPI Control Register 0
SPCR1 — QSPI Control Register 1
SPCR2 — QSPI Control Register 2
SPCR3 — QSPI Control Register 3
SPSR — QSPI Status Register
SWSR — Software Service Register
SYNCR — Clock Synthesizer Control Register
SYPCR — System Protection Control Register
TCNT — Timer Counter Register
TCTL1 — Timer Control Register 1
TCTL2 — Timer Control Register 2
TFLG1/TFLG2 — Timer Interrupt Flag Registers 1and 2
TI4/O5 — Input Capture 4/Output Compare 5 Register
TIC[1:3] — Input Capture Registers 1–3
TMSK1 — Timer Interrupt Mask Register 1
TMSK2 — Timer Interrupt Mask Register 2
TOC[1:4] — Output Compare Registers 1–4
TSTMSRA — Master Shift Register A
TSTMSRB — Master Shift Register B
TSTRC — Test Module Repetition Count
TSTSC — Test Module Shift Count

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 surveys the entire MC68HC16Z1 modular microcontroller. It lists features of each of the modules, shows device functional divisions and pinouts, 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 MC68HC16Z1 Features

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

3.1.1 System Integration Module

- External Bus Support
- Programmable Chip-Select Outputs
- System Protection Logic
- Watchdog Timer, Clock Monitor, and Bus Monitor
- Two 8-Bit Dual Function Ports
- One 7-Bit Dual Function Port
- Phase-Locked Loop (PLL) Clock System

3.1.2 CPU16

- 16-Bit Architecture
- Full Set of 16-Bit Instructions
- Three 16-Bit Index Registers
- Two 16-Bit Accumulators
- Control-Oriented Digital Signal Processing Capability
- 1 Megabyte of Program Memory and 1 Megabyte of Data Memory
- High-Level Language Support
- Fast Interrupt Response Time
- Background Debugging Mode
- Fully Static Operation

3.1.3 Analog-to-Digital Converter

- Eight Channels, Eight Result Registers
- Eight Automated Modes
- Three Result Alignment Modes
- One 8-Bit Digital Input Port

3.1.4 Queued Serial Module

- Enhanced Serial Communication Interface
- Queued Serial Peripheral Interface
- One 8-Bit Dual Function Port

3.1.5 General-Purpose Timer

- Two 16-Bit Free-Running Counters with Prescaler
- Three Input Capture Channels
- Four Output Compare Channels
- One Input Capture/Output Compare Channel
- One Pulse Accumulator/Event Counter Input
- Two Pulse Width Modulation Outputs
- One 8-Bit Dual Function Port
- Two Optional Discrete Inputs
- Optional External Clock Input

3.1.6 Standby Ram

- 1024-Byte Static Ram
- External Standby Voltage Supply Input

3.2 System Block Diagram and Pin Assignment Diagrams

Figure 3–1 is a functional diagram of the MC68HC16Z1. 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 is a pin assignment drawing based on a 132-pin plastic surface-mount package. Refer to **APPENDIX B MECHANICAL SPECIFICATIONS AND ORDERING INFORMATION** for package dimensions. All pin functions and signal names are shown in these drawings. See subsequent paragraphs in this section for pin and signal descriptions.

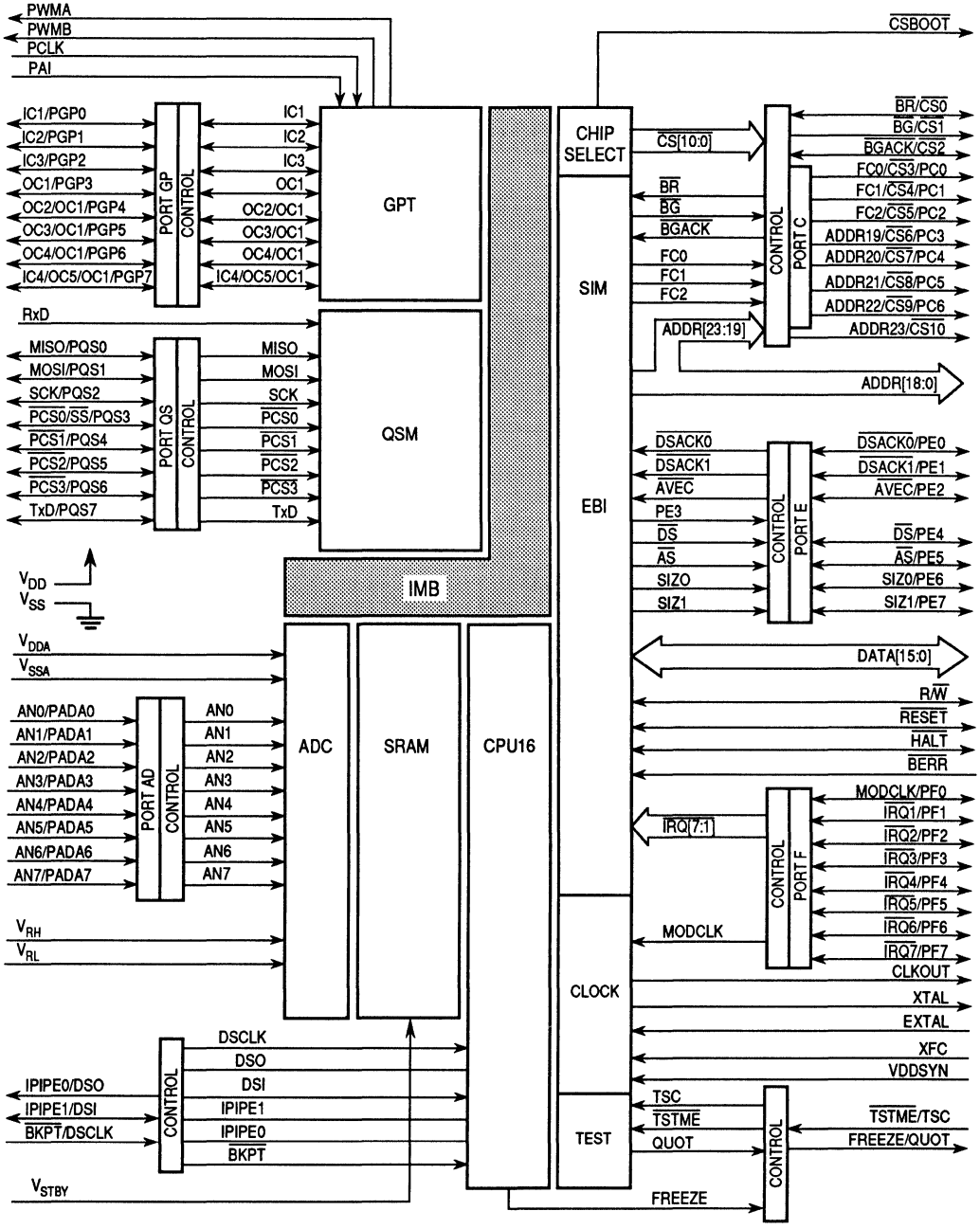


Figure 3–1. MC68HC16Z1 Block Diagram

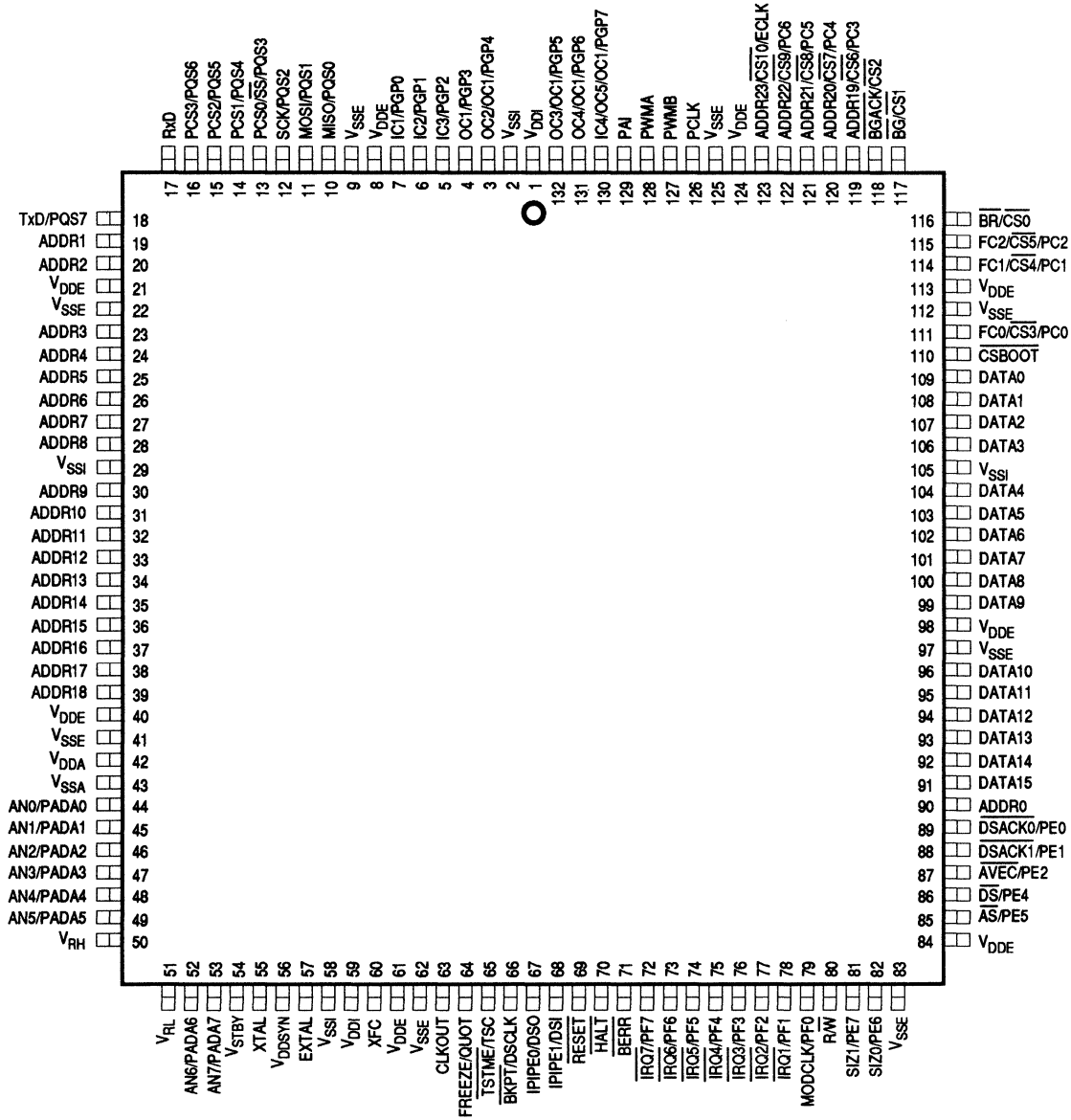


Figure 3–2. MC68HC16Z1 Pin Assignment for 132-Pin Package

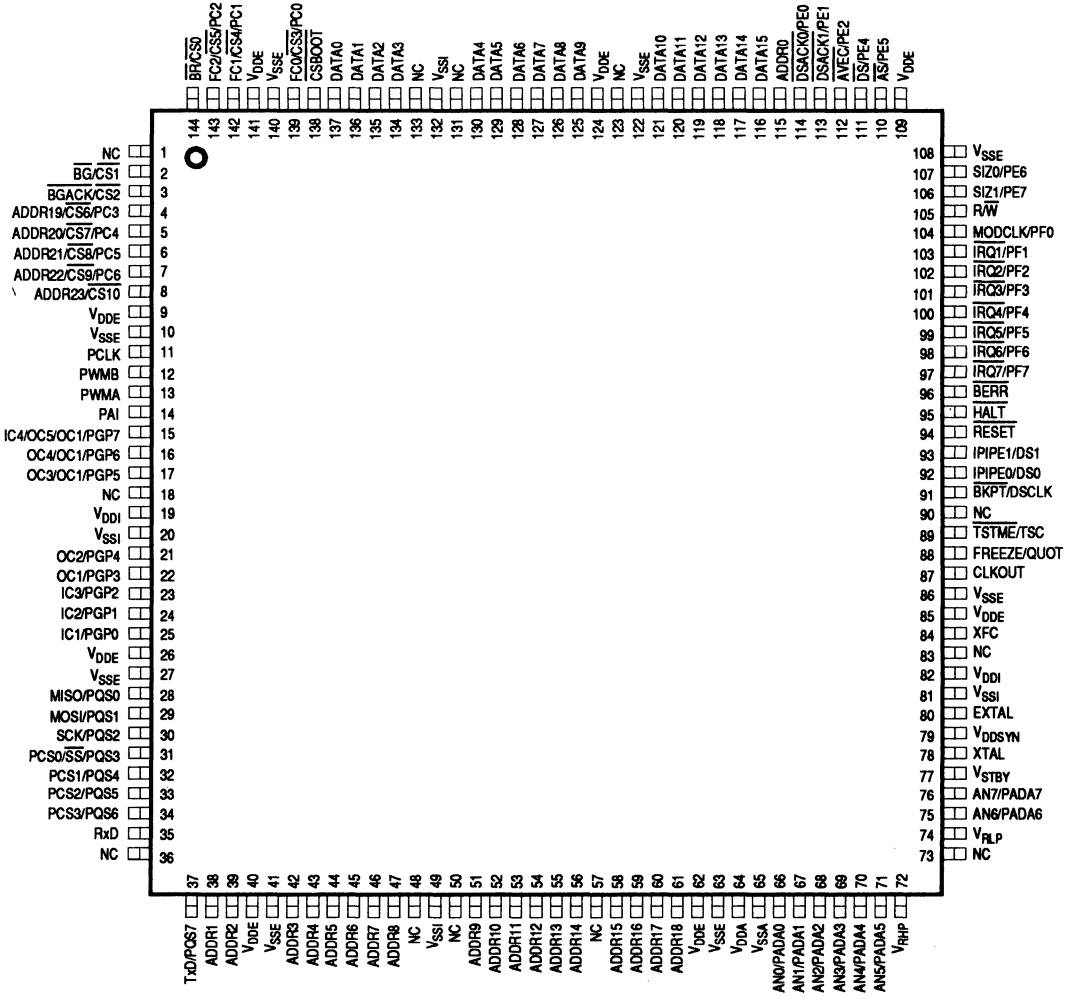


Figure 3–3. MC68HC16Z1 Pin Assignment for 144-Pin Package

3.3 Pin Descriptions

The tables below summarize functional characteristics of MC68HC16Z1 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 — 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.

3

Table 3–1. MC68HC16Z1 Driver Types

Type	I/O	Description
A	O	Output-only signals that are always driven; no external pullup required
Aw	O	Type A output with weak P-channel pullup 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. MC68HC16Z1 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	C[6:3]
ADDR[18:0]	A	Y	N	—	—
AN[7:0][1]	—	Y	N	I	ADA[7:0]
$\overline{AS}$	B	Y	N	I/O	E5
$\overline{AVEC}$	B	Y	N	I/O	E2
$\overline{BERR}$	B	Y	N	—	—
$\overline{BG}$/$\overline{CS1}$	B	—	—	—	—
$\overline{BGACK}$/$\overline{CS2}$	B	Y	N	—	—
$\overline{BKPT}$/DSCKL	—	Y	Y	—	—
$\overline{BR}$/$\overline{CS0}$	B	Y	N	O	Separate
CLKOUT	A	—	—	—	—
$\overline{CSBOOT}$	B	—	—	—	—
DATA[15:0][1]	AW	Y	N	—	—
$\overline{DS}$	B	Y	N	I/O	E4
$\overline{DSACK1}$	B	Y	N	I/O	E1
$\overline{DSACK0}$	B	Y	N	I/O	E0
DSI/IPIPE1	A	Y	Y	—	Separate
DSO/IPIPE0	A	—	—	—	Separate
EXTAL[2]	—	—	Special	—	—

Table 3–2. MC68HC16Z1 Pin Characteristics (Continued)

Pin Mnemonic	Output Driver	Input Synchronized	Input Hysteresis	Discrete I/O	Port Designation
FC[2:0]/$\overline{CS}$[5:3]	A	Y	N	O	C[2:0]
FREEZE/QUOT	A	—	—	—	—
$\overline{HALT}$	Bo	Y	N	—	—
IC4/OC5	A	Y	Y	I/O	GP4
IC[3:1]	A	Y	Y	I/O	GP[7:5]
$\overline{IRQ}$[7:1]	B	Y	Y	I/O	F[7:1]
MISO	Bo	Y	Y	I/O	QS0
MODCLK[1]	B	Y	N	I/O	F0
MOSI	Bo	Y	Y	I/O	QS1
OC[4:1]	A	Y	Y	I/O	GP[3:0]
PAI[3]	—	Y	Y	I	Separate
PCLK[3]	—	Y	Y	I	Separate
PCS0/$\overline{SS}$	Bo	Y	Y	I/O	QS3
PCS[3:1]	Bo	Y	Y	I/O	QS[6:4]
PWMA, PWMB[4]	A	—	—	O	Separate
R/$\overline{W}$	A	Y	N	—	—
$\overline{RESET}$	Bo	Y	Y	—	—
RXD	—	N	N	—	—
SCK	Bo	Y	Y	I/O	QS2
SIZ[1:0]	B	Y	N	I/O	E[7:6]
$\overline{TSTME}$/TSC	—	Y	Y	—	—
TXD	Bo	Y	Y	I/O	QS7
V_{RH}[5]	—	—	—	—	—
V_{RL}[5]	—	—	—	—	—
XFC[2]	—	—	—	Special	—
XTAL[2]	—	—	—	Special	—

NOTES
1. DATA[15:0] are synchronized during reset only. MODCLK, MCCI and ADC pins are synchronized only when used as input port pins.
2. EXTAL, XFC, and XTAL are clock reference connections.
3. PAI and PCLK can be used for discrete input, but are not part of an I/O port.
4. PWMA and PWMB can be used for discrete output, but are not part of an I/O port.
5. V_{RH} and V_{RL} are ADC reference voltage inputs.

Table 3–3. MC68HC16Z1 Power Connections

V_{STBY}	Standby RAM Power/Clock Synthesizer Power
V_{DDSYN}	Clock Synthesizer Power
V_{DDA}/V_{SSA}	A/D Converter 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 are a quick reference to MC68HC16Z1 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. MC68HC16Z1 Signal Characteristics

Signal Name	MCU Module	Signal Type	Active State
ADDR[23:0]	SIM	Bus	—
AN[7:0]	ADC	Input	—
$\overline{AS}$	SIM	Output	0
$\overline{AVEC}$	SIM	Input	0
$\overline{BERR}$	SIM	Input	0
$\overline{BG}$	SIM	Output	0
$\overline{BGACK}$	SIM	Input	0
$\overline{BKPT}$	CPU16	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	CPU16	Input	Serial Clock
DSI	CPU16	Input	(Serial Data)
DSO	CPU16	Output	(Serial Data)
EXTAL	SIM	Input	—
FC[2:0]	SIM	Output	—
FREEZE	SIM	Output	1
$\overline{HALT}$	SIM	Input/Output	0
IC[4:1]	GPT	Input	—
IPIPE0	CPU16	Output	—
IPIPE1	CPU16	Output	—
$\overline{IRQ[7:1]}$	SIM	Input	0
MISO	QSM	Input/Output	—
MODCLK	SIM	Input	—
MOSI	QSM	Input/Output	—
OC[5:1]	GPT	Output	—
PADA[7:0]	ADC	Input	(Port)
PAI	GPT	Input	—
PC[6:0]	SIM	Output	(Port)

Table 3–4. MC68HC16Z1 Signal Characteristics (Continued)

Signal Name	MCU Module	Signal Type	Active State
PE[7:0]	SIM	Input/Output	(Port)
PF[7:0]	SIM	Input/Output	(Port)
PGP[7:0]	GPT	Input/Output	(Port)
PQS[7:0]	QSM	Input/Output	(Port)
PCLK	GPT	Input	—
PCS[3:0]	QSM	Input/Output	—
PWMA, PWMB	GPT	Output	—
QUOT	SIM	Output	—
R/$\overline{W}$	SIM	Output	1/0
$\overline{RESET}$	SIM	Input/Output	0
RXD	QSM	Input	—
SCK	QSM	Input/Output	—
SIZ[1:0]	SIM	Output	—
$\overline{SS}$	QSM	Input	0
TSC	SIM	Input	—
$\overline{TSTME}$	SIM	Input	0
TXD	QSM	Output	—
V_{RH}	ADC	Input	—
V_{RL}	ADC	Input	—
XFC	SIM	Input	—
XTAL	SIM	Output	—

Table 3–5. MC68HC16Z1 Signal Function

Signal Name	Mnemonic	Function
Address Strobe	$\overline{AS}$	Indicates that a valid address is on the address bus
Autovector	$\overline{AVEC}$	Requests an automatic vector during interrupt acknowledge
Bus Error	$\overline{BERR}$	Indicates that a bus error has occurred
Bus Grant	$\overline{BG}$	Indicates that the MCU has relinquished the bus
Bus Grant Acknowledge	$\overline{BGACK}$	Indicates that an external device has assumed bus mastership
Breakpoint	$\overline{BKPT}$	Signals a hardware breakpoint to the CPU
Bus Request	$\overline{BR}$	Indicates that an external device requires bus mastership
Chip Selects	$\overline{CS[10:0]}$	Select external devices at programmed addresses
Boot Chip Select	$\overline{CSBOOT}$	Chip select for external boot startup ROM
Address Bus	ADDR[19:0]	20-bit address bus used by CPU16
Address Bus	ADDR[23:20]	4 MSB on IMB, test only, outputs follow ADDR19
ADC Analog Input	AN[7:0]	Inputs to ADC MUX
System Clockout	CLKOUT	System clock output
Data Bus	DATA[15:0]	16-bit data bus

Table 3–5. MC68HC16Z1 Signal Function (Continued)

Signal Name	Mnemonic	Function
Data Strobe	$\overline{DS}$	During a read cycle, indicates that an external device should place valid data on the data bus. During a write cycle, indicates that valid data is on the data bus
Halt	$\overline{HALT}$	Suspend external bus activity
Interrupt Request Level	$\overline{IRQ}$[7:1]	Provides an interrupt priority level to the CPU
Data and Size Acknowledge	$\overline{DSACK}$[1:0]	Provide asynchronous data transfers and dynamic bus sizing
Peripheral Chip Select	PCS[3:0]	QSPI peripheral chip selects
Reset	$\overline{RESET}$	System reset
Test Mode Enable	$\overline{TSTME}$	Hardware enable for SIM test mode
Development Serial In, Out, Clock	DSI, DSO, DSCLK	Serial I/O and clock for background debug mode
Crystal Oscillator	EXTAL, XTAL	Connections for clock synthesizer circuit reference; a crystal or an external oscillator can be used
Function Codes	FC[2:0]	Identify processor state and current address space
Freeze	FREEZE	Indicates that the CPU has entered background mode
Instruction Pipeline	PIPE[1:0]	Indicate instruction pipeline activity
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 ADA	PADA[7:0]	ADC digital input port signals
Port C	PC[6:0]	SIM digital output port signals
Port E	PE[7:0]	SIM digital I/O port signals
Port F	PF[7:0]	SIM digital I/O port signals
Port GP	PGP[7:0]	GPT 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
Read/Write	R/$\overline{W}$	Indicates the direction of data transfer on the bus
SCI Receive Data	RXD	Serial input to the SCI
QSPI Serial Clock	SCK	Clock output from QSPI in master mode; clock input to QSPI in slave mode
Size	SIZ[1:0]	Indicates the number of bytes to be transferred during a bus cycle
Slave Select	$\overline{SS}$	Causes serial transmission when QSPI is in slave mode; causes mode fault in master mode
Three-State Control	TSC	Places all output drivers in a high-impedance state
SCI Transmit Data	TXD	Serial output from the SCI
ADC Reference Voltage	V_{RH},V_{RL}	Provide precise reference for A/D conversion
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 design of modular microcontrollers. The modules in the MC68HC16Z1 communicate with one another and with external components via the IMB. Although the full IMB supports 24 address and 16 data lines, the CPU16 module in the MC68HC16Z1 uses only 16 data lines and 20 address lines. ADDR[23:20] are tied to ADDR19 when processor driven. ADDR[23:20] are brought out to pins for test purposes only.

3.6 System Memory Maps

Figures 3–3 through 3–5 are MC68HC16Z1 memory maps. Figure 3–3 shows IMB addresses of internal registers and the SRAM array. Figures 3–4 and 3–5 show system memory maps that use different external decoding schemes.

3.6.1 Internal Register Map

In Figure 3–3, 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 the MC68HC16Z1, 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). Since the CPU16 uses only ADDR[19:0], and ADDR[23:20] follow the logic state of ADDR19 when CPU driven, the CPU cannot access IMB addresses from $080000 to $F7FFFF. In order for the MCU to function correctly, MM must be set (Y must equal $F). If M is cleared, internal registers are mapped to base address $700000, and are inaccessible until a reset occurs. The SRAM array is positioned by a base address register in the SRAM CTRL block. Unimplemented blocks are mapped externally.

3.6.2 Pseudolinear Address Maps

Figures 3–4 and 3–5 both show the complete CPU16 pseudolinear address space. Address space can be split into physically distinct program and data spaces by decoding the MCU function code outputs. Figure 3–4 shows the memory map of a system that has combined program and data spaces. Figure 3–5 shows the memory map when MCU function code outputs are decoded.

Reset and exception vectors are mapped into bank 0 and cannot be relocated. The CPU16 program counter, stack pointer, and Z index register can be initialized to any address in pseudolinear memory, but exception vectors are limited to 16-bit addresses — to access locations outside of bank 0 during exception handler routines (including interrupt exceptions), a jump table must be used. 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, address space types, resets, and interrupts.

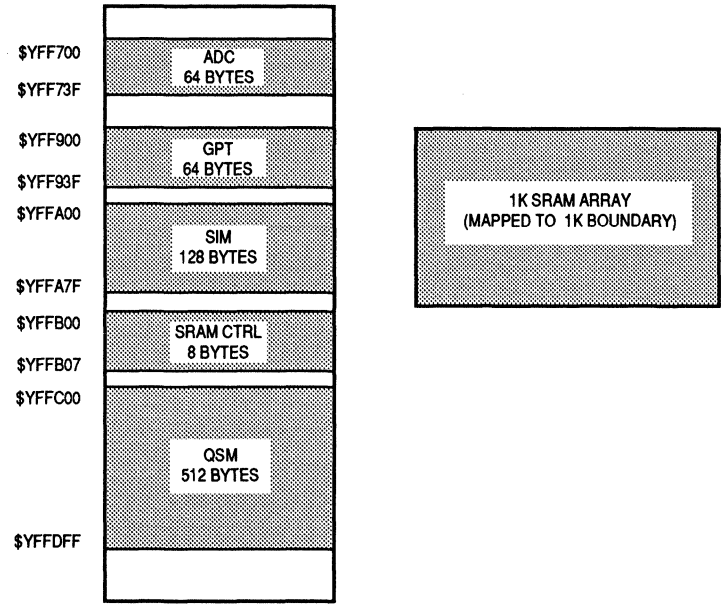


Figure 3–4. Internal Register Addresses

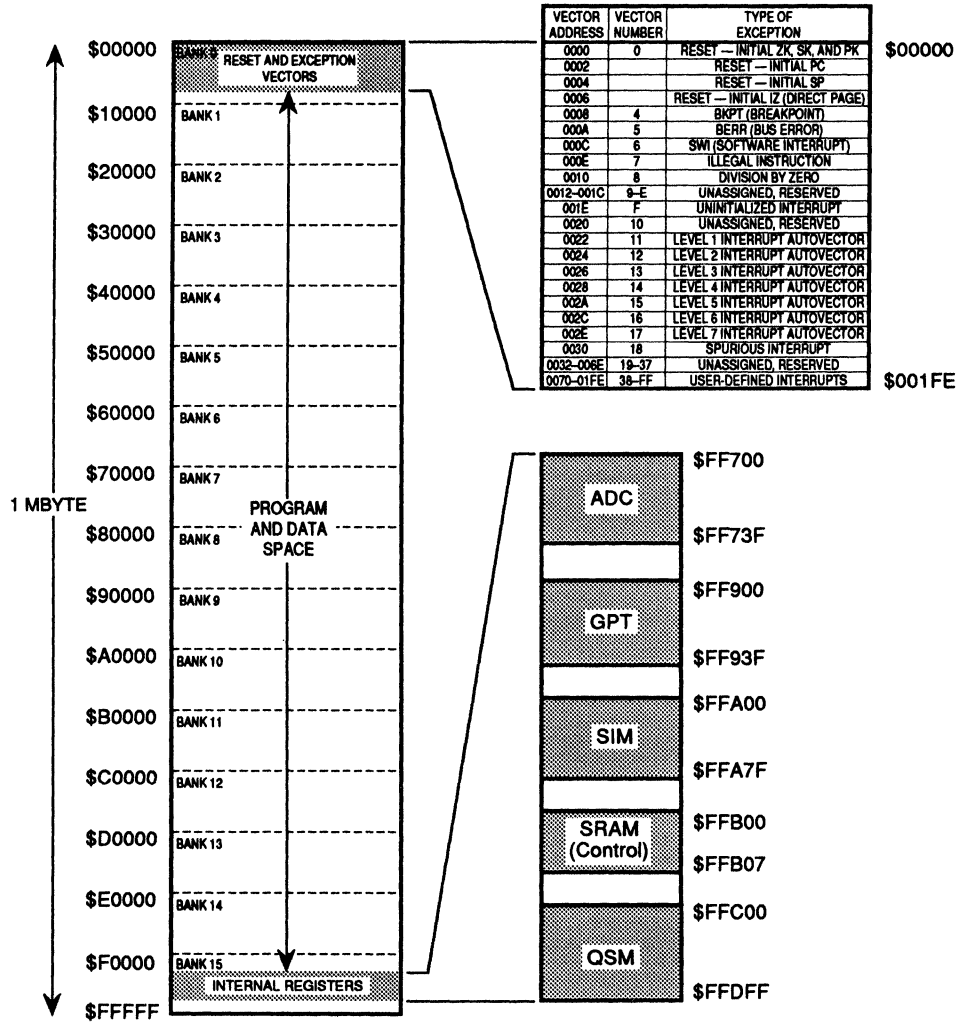


Figure 3–5. Pseudolinear Addressing With Combined Program and Data Spaces

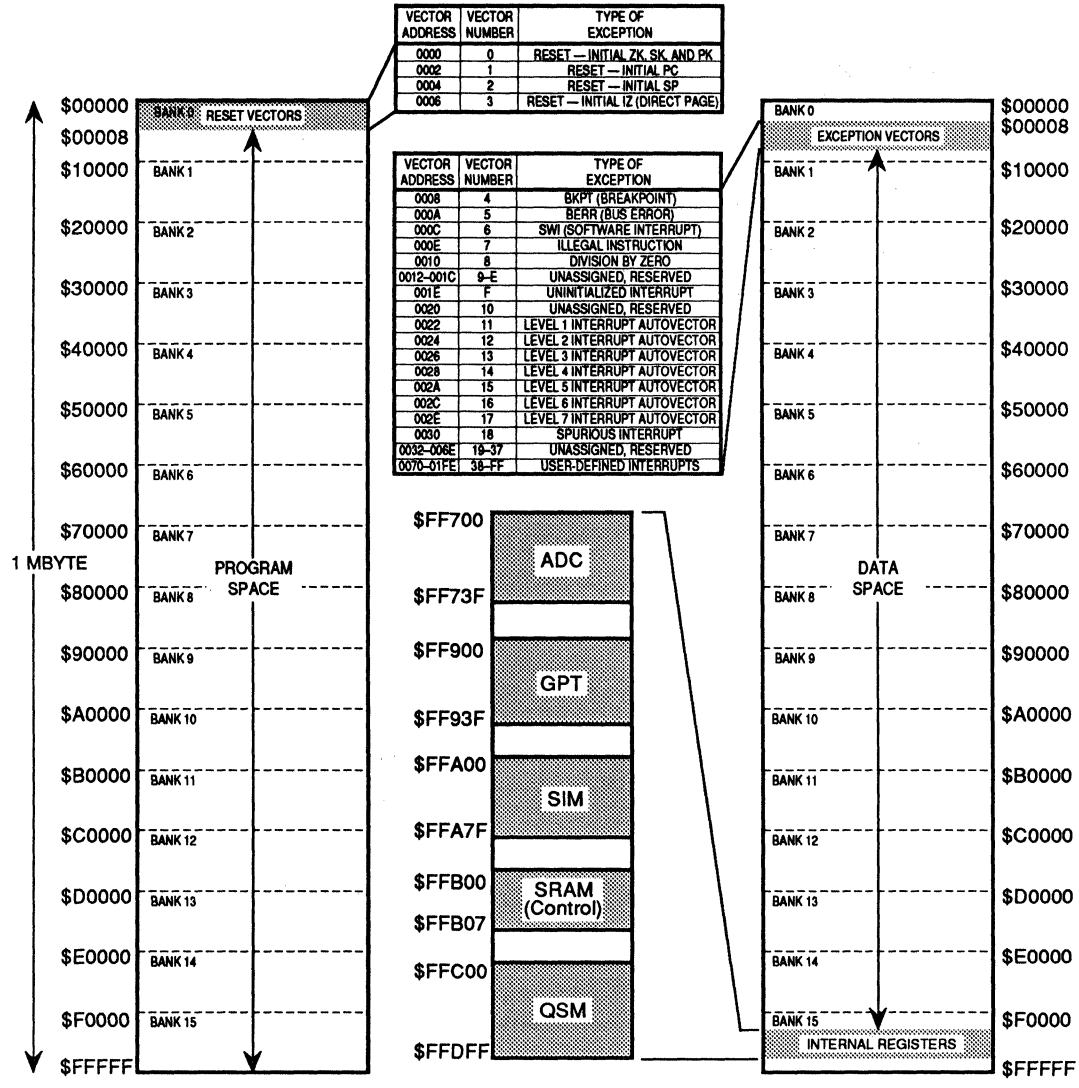


Figure 3–6. Pseudolinear Addressing With Separated Program and Data Spaces

3.7 System Reset

The following information is a concise reference only. MC68HC16Z1 system reset is a complex operation — to understand operation during and after reset, refer to **SECTION 4 SYSTEM INTEGRATION MODULE**, paragraph **4.5 Reset.**

3.7.1 System Reset Mode Selection

The logic states of certain data bus pins during reset determine system integration module 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	$\overline{\text{CS6}}$ $\overline{\text{CS7}}$–$\overline{\text{CS6}}$ $\overline{\text{CS8}}$–$\overline{\text{CS6}}$ $\overline{\text{CS9}}$–$\overline{\text{CS6}}$ $\overline{\text{CS10}}$–$\overline{\text{CS6}}$	ADDR19 ADDR[20:19] ADDR[21:19] ADDR[22:19] ADDR[23:19]
DATA8	$\overline{\text{DSACK0}}$, $\overline{\text{DSACK1}}$, $\overline{\text{AVEC}}$, $\overline{\text{DS}}$, $\overline{\text{AS}}$, $\overline{\text{SIZE}}$	PORTE
DATA9	$\overline{\text{IRQ7}}$–$\overline{\text{IRQ1}}$ 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, 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. Table 3–7 is a summary of module pin function following reset.

Table 3–7. Module Reset Pin Function

Module	Pin Mnemonic	Function
ADC	PADA[7:0]/AN[7:0]	DISCRETE INPUT
	V_{RH}	REFERENCE VOLTAGE
	V_{RL}	REFERENCE VOLTAGE
CPU	DSI/IPIPE1	DSI/IPIPE1
	DSO/IPIPE0	DSO/IPIPE0
	$\overline{\text{BKPT}}$/DSCKL	$\overline{\text{BKPT}}$/DSCKL
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

SECTION 4
SYSTEM INTEGRATION MODULE

The MC68HC16Z1 system integration module (SIM) consists of five functional blocks. Figure 4–1 is a block diagram of the SIM. Refer to **APPENDIX D REGISTER SUMMARY** for more information.

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.

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.

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.

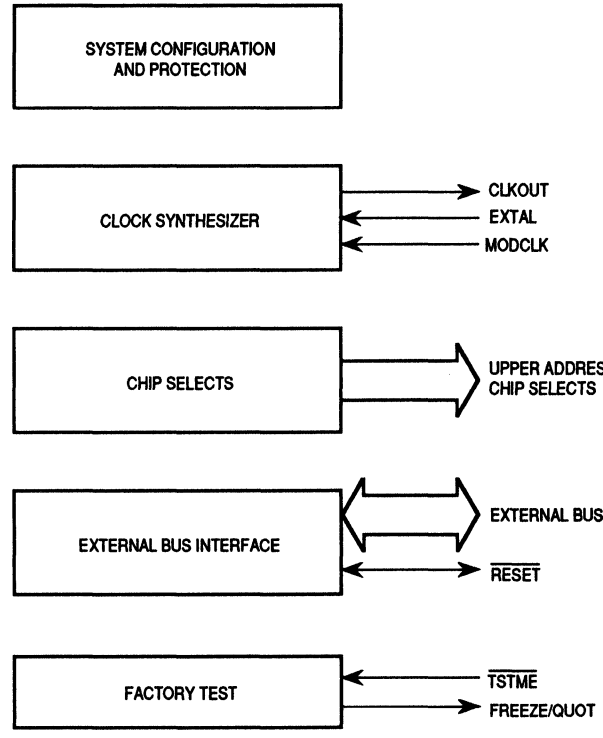


Figure 4–1. System Integration Module Block Diagram

4.1 System Configuration and Protection

This functional block controls MC68HC16Z1 module configuration, preserves reset status, monitors internal activity, and provides periodic interrupt generation. Figure 4–2 is a block diagram of the submodule.

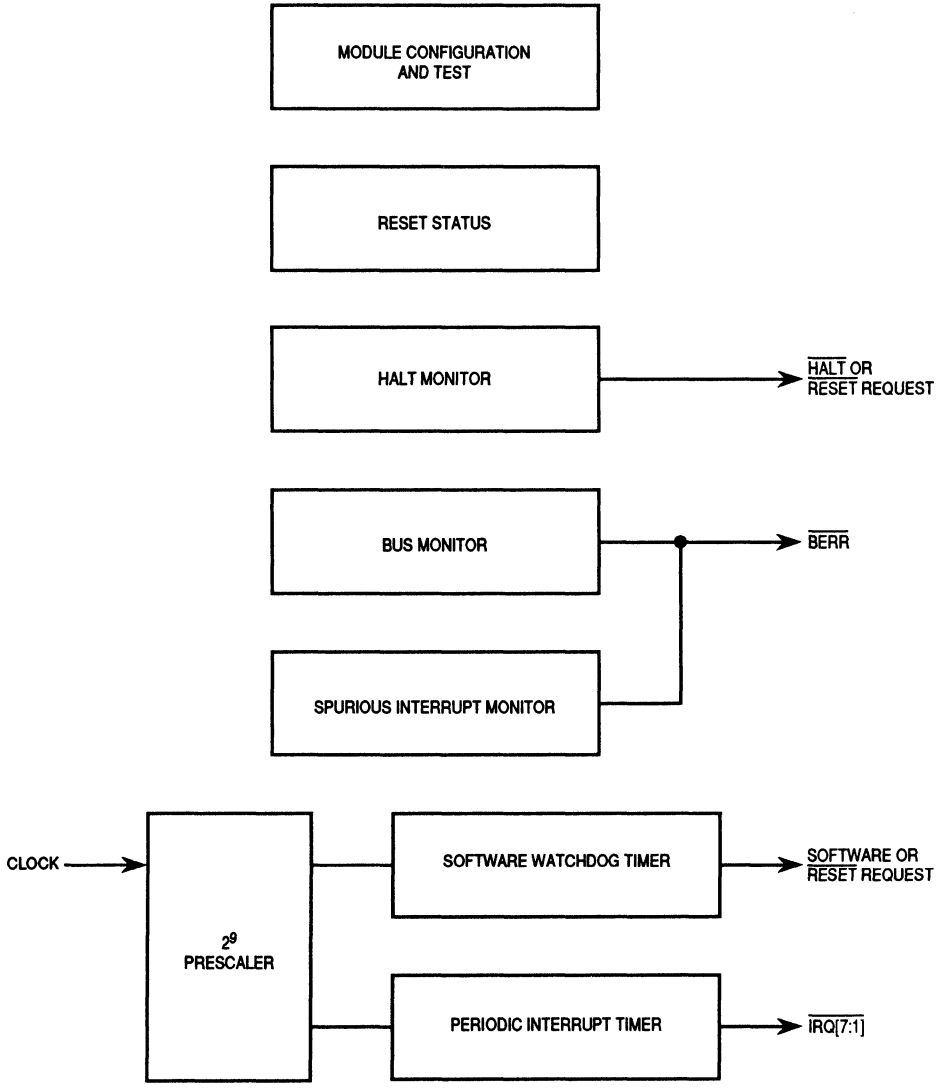


Figure 4–2. System Configuration and Protection

4.1.1 Module Mapping

Control registers for all the modules in the microcontroller are mapped into a 4-Kbyte block. The state of the module mapping (MM) bit in the SIM module configuration register (MCR) 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.

In the MC68HC16Z1, ADDR[23:20] follow the logic state of ADDR19 unless externally driven. MM corresponds to IMB ADDR23. If MM is cleared, the SIM maps IMB modules into address space $7FF000–$7FFFFF, which is inaccessible to the CPU16. Modules remain inaccessible until reset occurs. The reset state of MM is one, but the bit is one-time writable. Initialization software should make certain it remains set.

4.1.2 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 means of serial contention between IARB field bit values. Contention must take place whenever an interrupt request is acknowledged, even when there is only a single pending request. In order for contention to take place, an 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 CPU16 processes a spurious interrupt exception.

Because the SIM routes external interrupt requests to the CPU16, 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 — this prevents SIM interrupts from being discarded during initialization. Refer to **4.6 Interrupts** for a comprehensive discussion of interrupt arbitration.

4.1.3 Show Internal Cycles

A show cycle allows internal bus transfers to be externally monitored. The SHEN field in the MCR 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 **4.4.6.6 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

4.1.4 Factory Test Mode

The internal IMB can be slaved to an external master for direct module testing. This mode is reserved for factory testing. 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.

4.1.5 Register Access

Although the module configuration register contains a user/supervisor mode bit, SUPV, that is used to control access in some members of the modular microcontroller family, the MC68HC16Z1 always operates in the supervisor access mode. The state of SUPV has no meaning in the MC68HC16Z1.

4.1.6 Reset Status

The reset status register (RSR) shows internal MCU status during reset. Refer to **4.5.9 Reset Status Register** for more information.

4.1.7 Bus Monitor

The internal bus monitor checks for excessively long 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 response time is excessive.

$\overline{\text{DSACK}}$ 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 possible periods.

Table 4–2. Bus Monitor Period

BMT	Bus Monitor Timeout 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 CPU16 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 timeout period must be at least twice the number of clocks that a single byte access requires.

4.1.8 Halt Monitor

The halt monitor responds to an assertion of the $\overline{\text{HALT}}$ signal on the internal bus. Refer to **4.4.5.2 Double Bus Fault** for more information. Halt monitor reset can be inhibited by the halt monitor (HME) bit in SYPCR.

4.1.9 Spurious Interrupt Monitor

During interrupt exception processing, the CPU16 normally acknowledges an interrupt request, arbitrates among various sources of interrupt, 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 CPU16 to load the spurious interrupt exception vector into the program counter. The spurious interrupt monitor cannot be disabled. Refer to **4.6 Interrupts** for a comprehensive discussion of interrupts. Detailed information concerning interrupt exception processing is contained in **SECTION 5 CENTRAL PROCESSING UNIT**.

4.1.10 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:

a. Write $55 to SWSR.
b. Write $AA to SWSR.

Both writes must occur in the order listed prior to timeout, 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. 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 timeout period. Timeout period is given by the following equations.

Timeout Period = 1/(EXTAL Frequency/Divide Ratio)
or
Timeout Period = Divide Ratio/EXTAL Frequency

Table 4–4 gives 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 timeout 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.

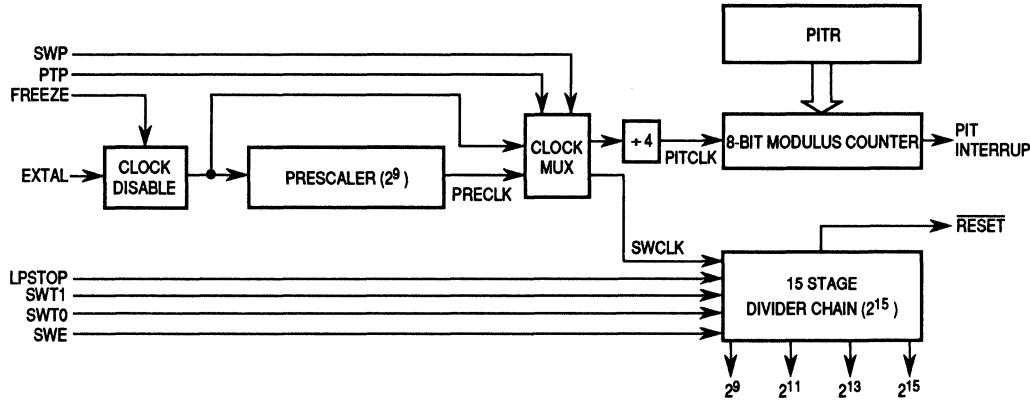


Figure 4–3.
Periodic Interrupt Timer and Software Watchdog Timer

4.1.11 Periodic Interrupt Timer

The periodic interrupt timer allows a user to generate 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 detailed information concerning 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. 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. 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.

PIT Period = [(PIT Modulus)(Prescaler value)(4)]/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 CPU16 interrupt priority mask to determine whether the interrupt is recognized. Table 4–6 shows priority of PIRQL values. Due to SIM hardware prioritization, a PIT interrupt is serviced before an external IRQ 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 is multiplied by two to form the vector offset, which is added to $0000 to obtain the address of the vector. Reset value of the PIV field is $0F, which generates the uninitialized interrupt vector.

4.1.12 Low-Power STOP Operation

When the CPU16 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 SYNCR, and the MC68HC16Z1 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. It continues to run at the same frequency as EXTAL during LPSTOP. A PIT interrupt can bring the MCU out of 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. To stop the periodic interrupt timer, PITR must be loaded with a zero value before the LPSTOP instruction is executed.

4.1.13 Freeze Operation

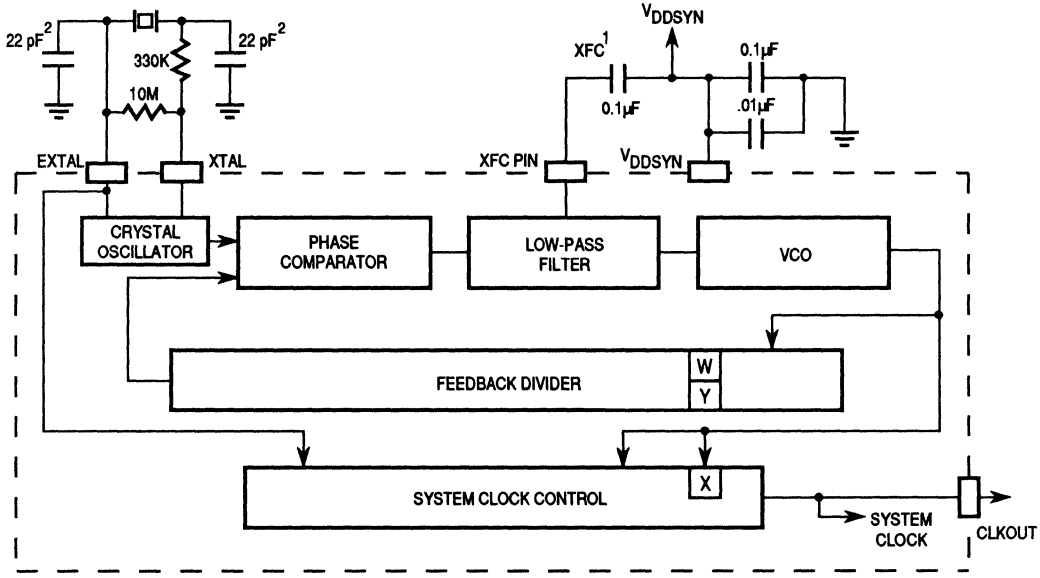
The FREEZE signal is used to halt MCU operations during debugging. FREEZE is asserted internally by the CPU16 if a breakpoint occurs while background mode is enabled. When FREEZE is asserted, only the bus monitor, software watchdog, and periodic interrupt timer are affected. The halt monitor and spurious interrupt monitor continue to operate normally. Setting the freeze bus monitor (FRZBM) bit in the MCR 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.

4.2 System Clock

The system clock in the SIM provides timing signals for the IMB modules and for an external peripheral bus. Because the MC68HC16Z1 is a fully static design, register and memory contents are not affected when clock rate changes. System hardware and software support changes in clock rate during operation.

The system clock signal can be generated in three ways. An internal phase-locked loop can synthesize the clock from either an internal or an external frequency source, or the clock signal can be input from an external source.

Figure 4–4 is a block diagram of the system clock.



1. MUST BE LOW-LEAKAGE CAPACITOR (INSULATION RESISTANCE 30,000 MΩ OR GREATER).
2. CAPACITANCE BASED ON A TEST CIRCUIT CONSTRUCTED WITH A DAISHINKU DMX-38 32.768 kHz CRYSTAL.

Figure 4–4. System Clock Block Diagram

4.2.1 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 — clock synthesizer control register SYNCR determines operating frequency and various modes 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.

A reference crystal must be connected between the EXTAL and XTAL pins in order to use the internal oscillator. A 32.768-kHz watch crystal is recommended — these crystals are readily available and inexpensive. MC68HC16Z1 clock synthesizer specifications (**APPENDIX A**, Table A–3, Clock Control Timing) are based upon a typical 32.768-kHz crystal.

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 either an external reference signal or an external system clock signal are applied, 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:

$$\text{Minimum external clock period} = \frac{\text{minimum external clock high/low time}}{50\% - \text{percentage variation of external clock input duty cycle}}$$

4.2.2 Clock Synthesizer Operation

A voltage controlled oscillator (VCO) generates the system clock signal. 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 internal crystal oscillator or from an external source. The comparator generates a control signal proportional to the difference in phase between its two inputs. The signal is low-pass filtered and used to correct VCO output frequency.

The synthesizer locks when VCO frequency is identical to reference 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.

The MC68HC16Z1 does not come out of reset state until the synthesizer locks. Crystal type, characteristic frequency, and layout of external oscillator circuitry affect lock time.

The low-pass filter requires an external low-leakage capacitor, typically 0.1 µF with an insulation resistance specification of 30,000 MΩ or greater, connected between the XFC and V_{DDSYN} pins.

V_{DDSYN} is used to power the clock circuits. A separate power source increases MCU noise immunity and can be used to run the clock when the MCU is powered down. A quiet power supply must be used as the V_{DDSYN} source. Adequate external bypass capacitors should be placed as close as possible to the V_{DDSYN} pin to assure stable operating frequency.

When the clock synthesizer is used, control register SYNCR determines operating frequency and various modes of operation. Because the CPU16 in the MC68HC16Z1 operates only in supervisor mode, SYNCR can be read or written at any time.

The SYNCR X bit controls a divide by two prescaler that is not in the synthesizer feedback loop. Setting X doubles clock speed without changing VCO speed — there is no VCO relock delay. The SYNCR W bit controls a 3-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 either W or Y value changes, there is a VCO relock delay (**APPENDIX A**, Table A–3, Clock Control Timing).

Clock frequency is determined by SYNCR bit settings as follows:

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

In order for the device to perform correctly, the clock frequency selected by the W, X, and Y bits must be within the limits specified for the MCU.

VCO frequency is determined by:

$$F_{VCO} = F_{SYSTEM}\,(2 - X)$$

The reset state of SYNCR ($3F00) produces a modulus-64 count — system frequency is 256 times reference frequency.

Table 4–7 shows multipliers for various combinations of SYNCR bits. The range of possible system frequencies can exceed the maximum specified system clock frequency. For instance, with a 32.768-kHz reference and a maximum system frequency of 16.78 MHz (**APPENDIX A**, Table A–3, Clock Control Timing), W and X must not both be set at any count modulus greater than Y = %001111. Table 4–8 shows available clock frequencies for a 16.78-MHz system with a 32.768-kHz reference.

Table 4–7. Clock Control Multipliers

To obtain clock frequency, find counter modulus in leftmost column, then multiply reference frequency by value in appropriate prescaler cell. Shaded cells are values that exceed specifications for the MC68HC16Z1.

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
000011	16	32	64	128
000100	20	40	80	160
000101	24	48	96	192
000110	28	56	112	224
000111	32	64	128	256
001000	36	72	144	288
001001	40	80	160	320
001010	44	88	176	352
001011	48	96	192	384
001100	52	104	208	416
001101	56	112	224	448
001110	60	120	240	480
001111	64	128	256	512
010000	68	136	272	544
010001	72	144	288	576
010010	76	152	304	608
010011	80	160	320	640
010100	84	168	336	672
010101	88	176	352	704
010110	92	184	368	736
010111	96	192	384	768
011000	100	200	400	800
011001	104	208	416	832
011010	108	216	432	864
011011	112	224	448	896
011100	116	232	464	928
011101	120	240	480	960
011110	124	248	496	992
011111	128	256	512	1024

Table 4–7. Clock Control Multipliers (Continued)

Modulus	Prescalers			
Y	[W:X] = 00	[W:X] = 01	[W:X] = 10	[W:X] = 11
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, find counter modulus in leftmost column, then look in appropriate prescaler cell. Empty cells represent values that exceed MC68HC16Z1 maximum system frequency specification.

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

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

Y	[W:X] = 00	[W:X] = 01	[W:X] = 10	[W:X] = 11
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
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

4.2.3 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). The operation of the external bus clock during low-power stop is described below. Refer to **4.7 Chip Selects** for more information concerning the external bus clock.

4.2.4 Low-Power Operation

Low-power operation is initiated by the CPU16. To reduce power consumption selectively, the CPU can set the STOP bits in each module configuration register. To reduce overall microcontroller power consumption to a minimum, 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 **4.4.4.2 LPSTOP Broadcast Cycles** 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, SIM clock control logic and the SIM clock signal (SIMCLK) continue to operate. The periodic interrupt timer and input logic for the $\overline{RESET}$ and $\overline{IRQ}$ 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 summarizes the effects of STSIM and STEXT with various clock sources. MODCLK value is the logic level on the MODCLK pin during the last reset prior to LPSTOP execution. Any clock in the off state is held low.

Table 4–9. Clock Control

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

4.2.5 Loss of Reference Signal

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

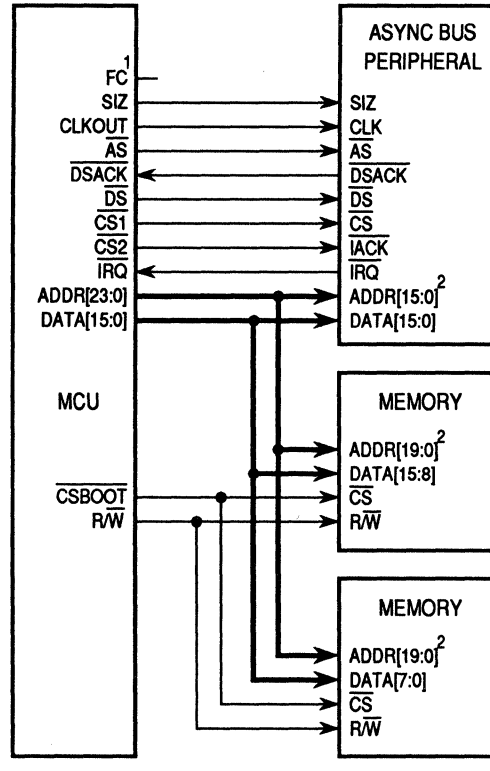
When RSTEN is cleared (default state out of reset), the clock synthesizer is forced into an operating condition referred to as limp mode. Limp mode frequency varies from device to device, but maximum limp frequency does not exceed one half maximum system clock when X = 0, or maximum system clock frequency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

4.3 External Bus Interface

The external bus interface (EBI) transfers information between the internal MCU bus and external devices. Figure 4–5 shows a basic system with external memory and peripherals.



1. CAN BE DECODED TO PROVIDE ADDITIONAL ADDRESS SPACE.
2. VARIES DEPENDING UPON PERIPHERAL MEMORY SIZE.

Figure 4–5. MC68HC16Z1 Basic System

The external bus has 24 address lines and 16 data lines. ADDR[19:0] are normal address outputs, ADDR[23:20] follow the output state of ADDR19.

A three-line handshaking interface performs external bus arbitration. The interface supports byte, word, and long-word transfers. The EBI performs dynamic sizing for data accesses.

The maximum number of bits transferred during an access is referred to as port width. Widths of 8 and 16 bits can be accessed by means of asynchronous bus cycles controlled by the data size (SIZ0 and SIZ1) and the data and size acknowledge ($\overline{\text{DSACK0}}$ and $\overline{\text{DSACK1}}$) 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 **4.7 Chip Selects** for more information.

4.3.1 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.

4.3.1.1 Address Bus

Bus signals ADDR[19:0] 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 $\overline{\text{AS}}$ is asserted.

4.3.1.2 Address Strobe

Address strobe ($\overline{\text{AS}}$) 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.

4.3.1.3 Data Bus

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

4.3.1.4 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.

4.3.1.5 Read/Write Signal

The read/write (R/$\overline{W}$) signal determines the direction of the transfer during a bus cycle. This signal changes state, when required, at the beginning of a bus cycle, and is valid while $\overline{AS}$ 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.

4.3.1.6 Size Signals

The 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

4.3.1.7 Function Codes

Function code signals FC[2:0] are generated by the CPU16. The function codes can be considered address extensions that designate which of eight external address spaces is accessed during a bus cycle.

Because the CPU16 always operates in supervisor mode (FC2 = 1), address spaces 0 to 3 are not used. 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
1	0	0	Reserved
1	0	1	Data Space
1	1	0	Program Space
1	1	1	CPU Space

4.3.1.8 Data and Size Acknowledge Signals

During normal bus transfers, external devices assert the data and size acknowledge signals ($\overline{\text{DSACK1}}$ and $\overline{\text{DSACK0}}$) 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 also indicate that an external device has successfully stored data and that the cycle may terminate. $\overline{\text{DSACK1}}$ and $\overline{\text{DSACK0}}$ can also be supplied internally by chip-select logic. Refer to **4.7 Chip Selects** for more information.

4.3.1.9 Bus Error Signal

The bus error ($\overline{\text{BERR}}$) signal is asserted in the absence of $\overline{\text{DSACK}}$ to indicate a bus error condition. It can also be asserted in conjunction with $\overline{\text{DSACK}}$ to indicate a bus error condition, provided it meets the appropriate timing requirements. Refer to **4.4.5 Bus Exception Control Cycles** for more information.

The internal bus monitor can be used to generate the $\overline{\text{BERR}}$ signal for internal and internal-to-external transfers. An external bus master must provide its own $\overline{\text{BERR}}$ generation and drive the $\overline{\text{BERR}}$ pin, because the internal $\overline{\text{BERR}}$ monitor has no information about transfers initiated by an external bus master. Refer to **4.4.6 Bus Arbitration** for more information.

4.3.1.10 Halt Signal

The halt signal ($\overline{\text{HALT}}$) can be asserted by an external device to cause single bus cycle operation. $\overline{\text{HALT}}$ is typically used for debugging purposes. When $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ are asserted simultaneously, the MC68HC16Z1 acts as though $\overline{\text{BERR}}$ alone is asserted. This may not be true of other modular microcontrollers. Refer to **4.4.5 Bus Exception Control Cycles** for more information.

4.3.1.11 Autovector Signal

The autovector signal (AVEC) can be used to terminate external interrupt acknowledge cycles. Assertion of AVEC causes the CPU16 to generate vector numbers to locate an interrupt handler routine. If it is continuously asserted, autovectors are generated for all external interrupt requests. AVEC is ignored during all other bus cycles. Refer to **4.6 Interrupts** for more information. AVEC for external interrupt requests can also be supplied internally by chip-select logic. Refer to **4.7 Chip Selects** for more information. The autovector function is disabled when there is an external bus master. Refer to **4.4.6 Bus Arbitration** for more information.

4

4.3.2 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 DSACK inputs, as shown in Table 4–12. Chip-select logic can generate data and size acknowledge signals for an external device. Refer to **4.7 Chip Selects** for more information.

Table 4–12.
Effect of DSACK Signals

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

For example, if the 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 DSACK signals to indicate the port width. For instance, a 16-bit device always returns 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 4–6. OP0 – OP3 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.

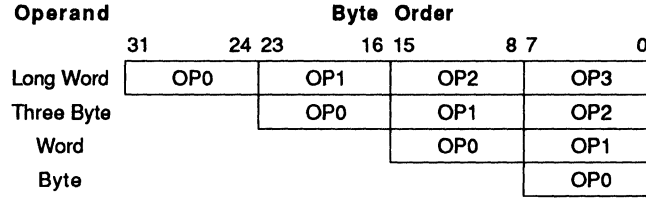


Figure 4–6. Operand Byte Order

4.3.3 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. Bear in mind the fact that ADDR[23:20] follow the state of ADDR19 in the MC68HC16Z1.

4.3.4 Misaligned Operands

CPU16 processor architecture uses a basic operand size of 16 bits. An operand is misaligned when it overlaps a word boundary. This is determined by the value of ADDR0. When ADDR0 = 0 (an even address), the address is on a word and byte boundary. When ADDR0 = 1 (an odd address), the address is

on a byte boundary only. A byte operand is aligned at any address; a word or long-word operand is misaligned at an odd address.

In the MC68HC16Z1, the largest amount of data that can be transferred by a single bus cycle is an aligned word. If the MCU transfers a long-word operand via a 16-bit port, the most significant operand word is transferred on the first bus cycle and the least significant operand word on a following bus cycle.

The CPU16 can perform misaligned word transfers. This capability makes it compatible with the MC68HC11 CPU. The CPU16 treats misaligned long-word transfers as two misaligned word transfers. Other modular microcontrollers have wider CPU architectures that support different long-word transfer cases.

4.3.5 Operand Transfer Cases

Table 4–13 summarizes how operands are aligned for various types of transfers. OPn entries are portions of a requested operand that are read or written during a bus cycle and are defined by SIZ1, SIZ0, and ADDR0 for that bus cycle. The following paragraphs discuss all the allowable transfer cases in detail.

Table 4–13. Operand Alignment

Transfer Case	SIZ1	SIZ0	ADDR0	$\overline{\text{DSACK1}}$	$\overline{\text{DSACK0}}$	DATA [15:8]	DATA [7:0]
Byte to 8-bit Port (Even/Odd)	0	1	X	1	0	OP0	(OP0)
Byte to 16-bit Port (Even)	0	1	0	0	X	OP0	(OP0)
Byte to 16-bit Port (Odd)	0	1	1	0	X	(OP0)	OP0
Word to 8-bit Port (Aligned)	1	0	0	1	0	OP0	(OP1)
Word to 8-bit Port (Misaligned)	1	0	1	1	0	OP0	(OP0)
Word to 16-bit Port (Aligned)	1	0	0	0	X	OP0	OP1
Word to 16-bit Port (Misaligned)	1	0	1	0	X	(OP0)	OP0
3 Byte to 8-bit Port (Aligned)[3]	1	1	0	1	0	OP0	(OP1)
3 Byte to 8-bit Port (Misaligned)[3]	1	1	1	1	0	OP0	(OP0)
3 Byte to 16-bit Port (Aligned)[4]	1	1	0	0	X	OP0	OP1
3 Byte to 16-bit Port (Misaligned)[3]	1	1	1	0	X	(OP0)	OP0
Long Word to 8-bit Port (Aligned)	0	0	0	1	0	OP0	(OP1)
Long Word to 8-bit Port (Misaligned)[4]	1	0	1	1	0	OP0	(OP0)
Long Word to 16-bit Port (Aligned)	0	0	0	0	X	OP0	OP1
Long Word to 16-bit Port (Misaligned)[4]	1	0	1	0	X	(OP0)	OP0

NOTES:
1. X in a column means that the state of the signal has no effect.
2. Operands in parentheses are ignored by the CPU16 during read cycles.
3. Three-byte transfer cases occur only as a result of an aligned long word to byte transfer.
4. The CPU16 treats misaligned long-word transfers as two misaligned word transfers.

4.3.5.1 Byte Operand to 8-Bit Port (ADDR0 = 0/1)

To initiate a transfer, the MCU places the desired address on the address bus and drives the size pins to indicate a single-byte operand.

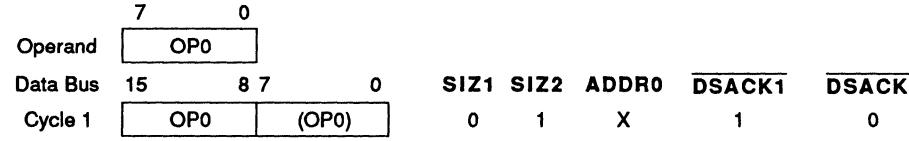


Figure 4–7. Byte Operand to 8-Bit Port (ADDR0 = 0, ADDR0 = 1)

4

For a read operation, the 8-bit peripheral responds by placing OP0 on DATA[15:8] and asserting $\overline{\text{DSACK0}}$. The MCU reads OP0 from DATA[15:8] and ignores DATA[7:0].

For a write operation, the MCU drives OP0 on both bytes of the data bus. The peripheral determines operand size and transfers the data to the specified address, then asserts $\overline{\text{DSACK0}}$ to terminate the bus cycle.

4.3.5.2 Byte Operand to 16-Bit Port, Even (ADDR0 = 0)

To initiate transfer, the MCU places the desired address on the address bus and drives the size pins to indicate a single-byte operand.

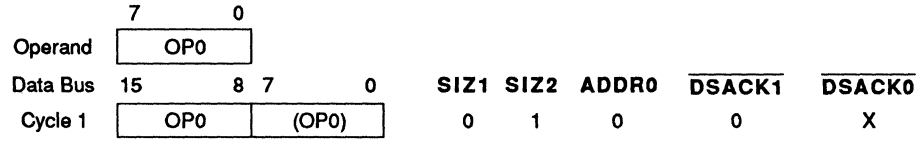


Figure 4–8. Byte Operand to 16-Bit Port, Even (ADDR0 = 0)

4

For a read operation, the 16-bit peripheral responds by placing OP0 on DATA[15:8] and asserting $\overline{\text{DSACK1}}$. The MCU reads the data from DATA[15:8] and ignores DATA[7:0].

For a write operation, the MCU drives OP0 on both bytes of the data bus. The peripheral determines operand size and transfers the data to the specified address, then asserts $\overline{\text{DSACK1}}$ to terminate the bus cycle.

4.3.5.3 Byte Operand to 16-Bit Port, Odd (ADDR0 = 1)

To initiate transfer, the MCU places the desired address on the address bus and drives the size pins to indicate a single-byte operand.

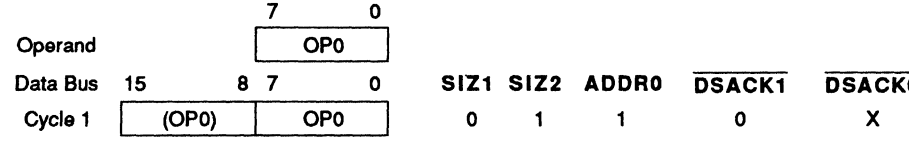


Figure 4–9. Byte Operand to 16-Bit Port, Odd (ADDR0 = 1)

For a read operation, the 16-bit peripheral responds by placing OP0 on DATA[7:0] and asserting $\overline{\text{DSACK1}}$. The MCU then reads the data from DATA[7:0] and ignores DATA[15:8].

For a write operation, the MCU drives OP0 on both bytes of the data bus. The peripheral determines operand size and transfers OP0 to the specified address, then asserts $\overline{\text{DSACK1}}$ to terminate the bus cycle.

4.3.5.4 Word Operand to 8-Bit Port, Aligned

To initiate transfer, the MCU places the desired address on the address bus and drives the size pins to indicate a word operand.

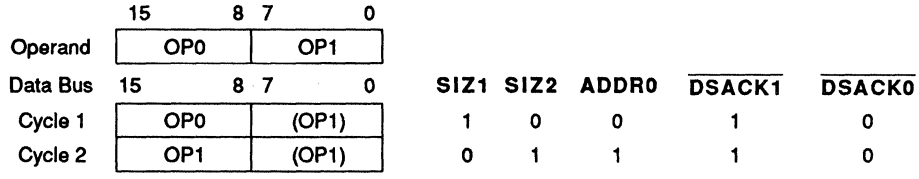


Figure 4–10. Word Operand to 8-Bit Port, Aligned

For a read operation, the 8-bit peripheral responds by placing OP0 on DATA[15:8] and asserting $\overline{\text{DSACK0}}$. The MCU reads OP0 from DATA[15:8] and ignores DATA[7:0], then decrements the transfer size counter, increments the address, and performs a byte operand to 8-bit port transfer of OP1.

For a write operation, the MCU drives OP0 on DATA[15:8] and OP1 on DATA[7:0]. The 8-bit peripheral transfers OP0 from DATA[15:8] to the specified address, then asserts $\overline{\text{DSACK0}}$ to indicate that the first byte has been transferred. The MCU then decrements the transfer size counter, increments the address, and performs a byte operand to 8-bit port transfer of OP1.

4.3.5.5 Word Operand to 8-Bit Port, Misaligned

To initiate transfer, the MCU places the desired address on the address bus and drives the size pins to indicate a word operand.

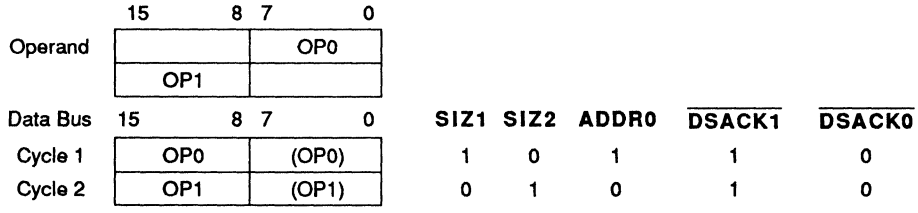


Figure 4–11. Word Operand to 8-Bit Port, Misaligned

For a read operation, the 8-bit peripheral responds by placing OP0 on DATA[15:8] and asserting $\overline{\text{DSACK0}}$. The MCU reads the upper operand byte from DATA[15:8] and ignores DATA[7:0]. The MCU then decrements the transfer size counter, increments the address, and performs a byte operand to 8-bit port transfer of OP1.

For a write operation, the MCU drives OP0 on DATA[15:8] and OP1 on DATA[7:0]. The 8-bit peripheral transfers OP0 to the specified address, then asserts $\overline{\text{DSACK0}}$ to indicate that the first byte of word data has been received. The MCU then decrements the transfer size counter, increments the address, and performs a byte operand to 8-bit port transfer of OP1.

4.3.5.6 Word Operand to 16-Bit Port, Aligned

To initiate transfer, the MCU places the desired address on the address bus and drives the size pins to indicate a word operand.

Figure 4–12. Word Operand to 16-Bit Port, Aligned

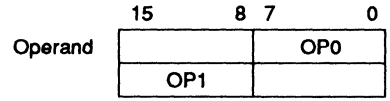
4

For a read operation, the peripheral responds by placing OP0 on DATA[15:8] and OP1 on DATA[7:0], then asserts $\overline{\text{DSACK1}}$ to indicate a 16-bit port. When $\overline{\text{DSACK1}}$ is asserted, the MCU reads DATA[15:8] and terminates the cycle.

For a write operation, the MCU drives the word operand on DATA[15:0]. The peripheral device then reads the entire operand from DATA[15:0] and asserts $\overline{\text{DSACK1}}$ to terminate the bus cycle.

4.3.5.7 Word Operand to 16-Bit Port, Misaligned

To initiate transfer, the MCU places the desired address on the address bus and drives the size pins to indicate a word operand.



Data Bus	15 — 8	7 — 0	SIZ1	SIZ2	ADDR0	$\overline{\text{DSACK1}}$	$\overline{\text{DSACK0}}$
Cycle 1	(OP0)	OP0	1	0	1	0	X
Cycle 2	OP1	(OP1)	0	1	0	0	X

Figure 4–13. Word Operand to 16-Bit Port, Misaligned

For a read operation, the peripheral responds by placing OP0 on DATA[7:0] and asserting $\overline{\text{DSACK1}}$ to indicate a 16-bit port. When $\overline{\text{DSACK1}}$ is asserted, the MCU reads OP0 from DATA[7:0], decrements the transfer size counter, increments the address, and performs a byte operand from 8-bit port transfer to acquire OP1.

For a write operation, the MCU first drives OP0 on DATA[7:0] and duplicates it on DATA[15:8]. The peripheral device reads OP0 from DATA[7:0] and asserts $\overline{\text{DSACK1}}$. The MCU decrements the transfer size counter, increments the address, and performs a byte operand to 8-bit port transfer of OP1.

4.3.5.8 Long-Word Operand to 8-Bit Port, Aligned

The MCU drives the address bus with the desired address and the size pins to indicate a long word operand.

	15 8	7 0
Operand	OP0	OP1
	OP2	OP3

Data Bus	15 8	7 0	SIZ1	SIZ2	ADDR0	$\overline{\text{DSACK1}}$	$\overline{\text{DSACK0}}$
Cycle 1	OP0	(OP1)	0	0	0	1	0
Cycle 2	OP1	(OP1)	1	1	1	1	0
Cycle 3	OP2	(OP2)	1	0	0	1	0
Cycle 4	OP3	(OP3)	0	1	1	1	0

Figure 4–14. Long-Word Operand to 8-Bit Port, Aligned

For a read operation, the peripheral places OP0 on DATA[15:8] and asserts $\overline{\text{DSACK0}}$ to indicate an 8-bit port. The MCU reads OP0 from DATA[15:8] and ignores DATA[7:0]. The MCU then decrements the transfer size counter, increments the address, initiates a new cycle, and reads OP1 from DATA[15:8]. The process repeats for OP2 and OP3.

For a write operation, the MCU drives OP0 and OP1 on DATA[15:0]. The peripheral device then reads only OP0 from DATA[15:8] and asserts $\overline{\text{DSACK0}}$ to indicate an 8-bit port. The MCU then decrements the transfer size counter, increments the address, and writes OP1 to DATA[15:8]. The process repeats for OP2 and OP3.

4.3.5.9 Long-Word Operand to 8-Bit Port, Misaligned

The CPU16 treats misaligned long words as two misaligned words. The MCU drives the address bus with the desired address and the size pins to indicate a word operand.

	15 8	7 0
Operand		OP0
	OP1	OP2
	OP3	

Data Bus	15 8	7 0	SIZ1	SIZ2	ADDR0	$\overline{\text{DSACK1}}$	$\overline{\text{DSACK0}}$
Cycle 1	OP0	(OP0)	1	0	1	1	0
Cycle 2	OP1	(OP1)	0	1	0	1	0
Cycle 3	OP2	(OP2)	1	0	1	1	0
Cycle 4	OP3	(OP3)	0	1	0	1	0

Figure 4–15. Long-Word Operand to 8-Bit Port, Misaligned

For a read operation, the 8-bit peripheral responds by placing OP0 on DATA[15:8] and asserting $\overline{\text{DSACK0}}$. The MCU reads OP0 from DATA[15:8] and ignores DATA[7:0]. The MCU then decrements the transfer size counter, increments the address, and performs a byte operand to 8-bit port transfer of OP1. One misaligned word has been read — the process repeats for the second word (OP2 and OP3).

For a write operation, the MCU drives OP0 on DATA[15:8] and OP1 on DATA [7:0]. The 8-bit peripheral transfers OP0 to the specified address, then asserts $\overline{\text{DSACK0}}$ to indicate that the first byte of word data has been received. The MCU then decrements the transfer size counter, increments the address, and performs a byte operand to 8-bit port transfer of OP1. One misaligned word has been written — the process repeats for the second word (OP2 and OP3).

4.3.5.10 Long-Word Operand to 16-Bit Port, Aligned

The MCU drives the address bus with the desired address and drives the size pins to indicate a long-word operand.

	15 8	7 0
Operand	OP0	OP1
	OP2	OP3

Data Bus	15 8	7 0	SIZ1	SIZ2	ADDR0	$\overline{\text{DSACK1}}$	$\overline{\text{DSACK0}}$
Cycle 1	OP0	OP1	0	0	0	0	X
Cycle 2	OP2	OP2	1	0	0	0	X

Figure 4–16. Long-Word Operand to 16-Bit Port, Aligned

For a read operation, the 16-bit peripheral responds by placing OP0 on DATA[15:8] and OP1 on DATA[7:0], then asserts $\overline{\text{DSACK1}}$ to indicate a 16-bit port. The MCU reads OP0 and OP1 from DATA[15:0]. The process repeats for OP2 and OP3.

For a write operation, the MCU drives OP0 on DATA[15:8] and OP1 on DATA[7:0]. The peripheral device reads OP0 and OP1 from DATA[15:0] and asserts $\overline{\text{DSACK1}}$ to indicate a 16-bit port. The process repeats for OP2 and OP3.

4.3.5.11 Long-Word Operand to 16-Bit Port, Misaligned

The CPU16 treats misaligned long-word transfers as two misaligned word transfers. The MCU drives the address bus with the desired address and drives the size pins to indicate a word operand.

	15 8	7 0
Operand		OP0
	OP1	OP2
	OP3	

Data Bus	15 8	7 0	SIZ1	SIZ2	ADDR0	$\overline{\text{DSACK1}}$	$\overline{\text{DSACK0}}$
Cycle 1	(OP0)	OP0	1	0	1	0	X
Cycle 2	OP1	(OP1)	0	1	0	0	X
Cycle 3	(OP2)	OP2	1	0	1	0	X
Cycle 4	OP3	(OP3)	0	1	0	0	X

Figure 4–17. Long-Word Operand to 16-Bit Port, Misaligned

For a read operation, the peripheral responds by placing OP0 on DATA[7:0] and asserting $\overline{\text{DSACK1}}$ to indicate a 16-bit port. When $\overline{\text{DSACK1}}$ is asserted, the MCU reads OP0 from DATA[7:0], decrements the transfer size counter, increments the address, and performs a byte operand from 8-bit port transfer to acquire OP1. The process is repeated for OP2 and OP3.

For a write operation, the MCU first drives OP0 on DATA[7:0] and duplicates it on DATA[15:8]. The peripheral device reads OP0 from DATA[7:0] and asserts $\overline{\text{DSACK1}}$. The MCU decrements the transfer size counter, increments the address, and performs a byte operand to 8-bit port transfer of OP1. The process is repeated for OP2 and OP3.

4.4 Bus Operation

Internal microcontroller modules are typically accessed in 2 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 **4.4.2 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 **4.4.3 Fast Termination Cycles** and **4.7 Chip Selects** for more information. Bus control signal timing as well as chip-select signal timing are specified in **APPENDIX A ELECTRICAL CHARACTERISTICS**.

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

4.4.1 Synchronization to CLKOUT

External devices connected to the MCU bus can operate at clock frequencies different from that of the MCU so long as they 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, labelled {S0, S1, S2,..., SN} in the appropriate timing diagrams. 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 DSACK for the required window around the falling edge of S2 and obeys the bus protocol by maintaining DSACK and BERR or HALT until and throughout the clock edge that negates AS (with the appropriate asynchronous input hold time), no wait states are inserted. The bus cycle runs at the maximum speed of three clocks per cycle.

To assure proper operation in a system synchronized to CLKOUT, when BERR (or BERR and HALT) is asserted after DSACK, BERR (or BERR and HALT) assertion must satisfy the appropriate data-in setup and hold times prior to the falling edge of the clock cycle after DSACK is recognized.

4.4.2 Regular Bus Cycles

The following discussion pertains to cycles using external bus control logic. Refer to **4.4.3 Fast Termination Cycles** for information concerning 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 **4.3.2 Dynamic Bus Sizing**). 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), then asserts a DSACK[1:0] combination that indicates port size.

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

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

NOTE

The SIM bus monitor asserts 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 CPU16 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.

4.4.2.1 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. Refer to **4.3.2 Dynamic Bus Sizing** and **4.3.4 Misaligned Operands** for more information. Figure 4–18 is a flowchart of a word read cycle.

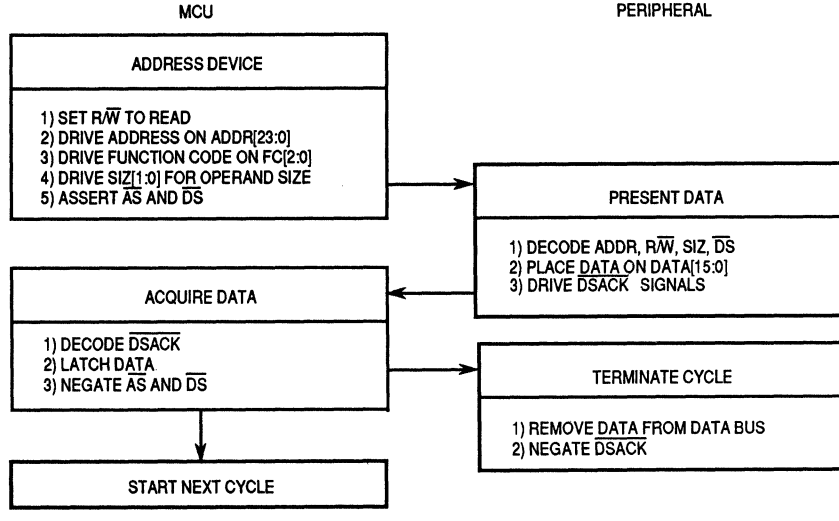


Figure 4–18. Word Read Cycle Flowchart

State 0 (S0) — The read cycle starts. The MCU places an address on ADDR [23:0] and function codes on FC[2:0]. In the MC68HC16Z1, ADDR[23:20] always follow the state of ADDR19, and FC2 is always equal to one. The MCU drives R/$\overline{W}$ high for a read cycle. Size signals SIZ[1:0] become valid, indicating the number of bytes to be read.

State 1 (S1) — The MCU asserts $\overline{AS}$ indicating that the address on the address bus is valid. The MCU also asserts $\overline{DS}$, indicating that data can be placed on the bus.

State 2 (S2) — External logic decodes ADDR[23:0], FC[1:0], R/$\overline{W}$, SIZ[1:0], and $\overline{DS}$. One or both of DATA[15:8] and DATA[7:0] are selected, and the responding device places data on that portion of the bus. Concurrently, the device asserts the appropriate $\overline{DSACK}$ signals. If the MCU is to latch the data in minimum cycle time, at least one $\overline{DSACK}$ signal must change state by the end of S2 in order to satisfy asynchronous input setup time requirements. If wait states are to be inserted, both $\overline{DSACK1}$ and $\overline{DSACK0}$ must remain negated throughout the asynchronous input setup and hold times at the end of S2.

State 3 (S3) — When a change in one or both of the $\overline{DSACK}$ signals has been recognized, the MCU latches data from the bus on the next falling edge of the clock (S4), and the cycle terminates (S5). If neither $\overline{DSACK}$ signal changes state by the start of S3, the MCU inserts wait states instead of proceeding to S4 and S5. While wait states are added, the MCU continues to sample the $\overline{DSACK}$ signals on falling edges of the clock until a change in one or more is recognized. In effect, S3 repeats until a change in the $\overline{DSACK}$ signals is detected.

State 4 (S4) — The MCU latches data on the falling edge at the end of S4.

State 5 (S5) — The MCU negates $\overline{AS}$ and $\overline{DS}$, but holds the address valid to provide address hold time for memory systems. R/$\overline{W}$, SIZ[1:0], and FC[2:0] also remain valid throughout S5. The external device must maintain data and assert the $\overline{DSACK}$ signals until it detects the negation of either $\overline{AS}$ or $\overline{DS}$ it must remove the data and negate $\overline{DSACK}$ within approximately one clock period after sensing the negation of $\overline{AS}$ or $\overline{DS}$. Signals that remain asserted beyond this limit can be prematurely detected during the next bus cycle.

4.4.2.2 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 **4.3.2 Dynamic Bus Sizing** and **4.3.4 Misaligned Operands** for more information. Figure 4–19 is a flowchart of a write-cycle operation for a word transfer.

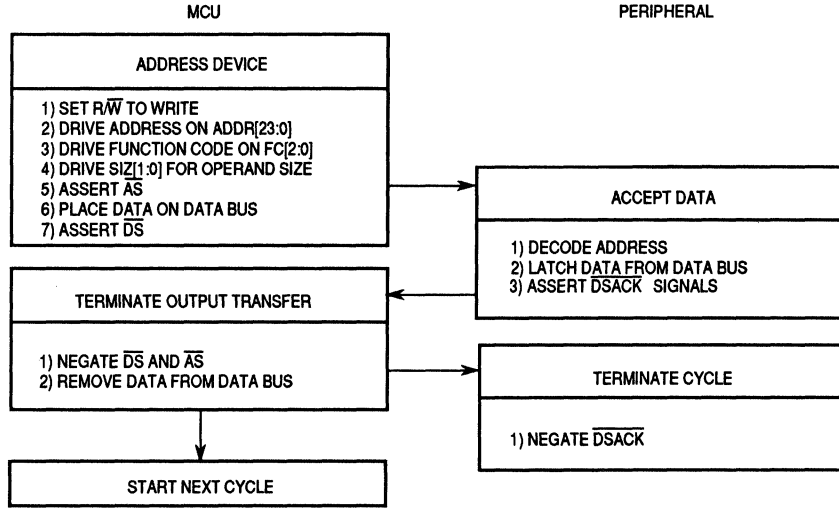


Figure 4–19. Write Cycle Flowchart

State 0 (S0) — The MCU places an address on ADDR[23:0] and function codes on FC[2:0]. In the MC68HC16Z1, ADDR[23:20] always follow the state of ADDR19, and FC2 is always equal to one. The MCU drives R/$\overline{W}$ low for a write cycle. Size signals SIZ[1:0] become valid, indicating the number of bytes to be written.

State 1 (S1) — The MCU asserts $\overline{AS}$, indicating that the address on the address bus is valid.

State 2 (S2) — The MCU places the data to be written onto DATA[15:0], then begins to sample the $\overline{\text{DSACK}}$ signals. External logic decodes ADDR[23:0], FC[1:0], R/$\overline{\text{W}}$, SIZ[1:0], and $\overline{\text{AS}}$. One or both of DATA[15:8] and DATA[7:0] are selected, and appropriate $\overline{\text{DSACK}}$ signals are asserted. If the cycle is to end in minimum time, the MCU must recognize a change in at least one $\overline{\text{DSACK}}$ signal by the end of S2. If wait states are to be inserted, both $\overline{\text{DSACK1}}$ and $\overline{\text{DSACK0}}$ must remain negated throughout the asynchronous input setup and hold times at the end of S2.

State 3 (S3) — The MCU asserts $\overline{\text{DS}}$ to indicate that data is stable on the data bus, and the selected peripheral latches the data. When a change in one or both of the $\overline{\text{DSACK}}$ signals has already been recognized, S4 elapses, and the cycle terminates during S5. If neither $\overline{\text{DSACK}}$ signal changes state by the start of S3, the MCU inserts wait states instead of proceeding to S4 and S5. While wait states are added, the MCU continues to sample the $\overline{\text{DSACK}}$ signals on falling edges of the clock until a change in one or more is recognized. In effect, S3 repeats until a change in the $\overline{\text{DSACK}}$ signals is detected.

State 4 (S4) — The MCU issues no new control signals during S4.

State 5 (S5) — The MCU negates $\overline{\text{AS}}$ and $\overline{\text{DS}}$, but holds the address and data valid to provide address hold time for memory systems. R/$\overline{\text{W}}$, SIZ[1:0], and FC[2:0] also remain valid throughout S5. The external device must assert the $\overline{\text{DSACK}}$ signals until it detects the negation of either $\overline{\text{AS}}$ or $\overline{\text{DS}}$. It must negate $\overline{\text{DSACK}}$ within approximately one clock period after sensing the negation of $\overline{\text{AS}}$ or $\overline{\text{DS}}$. Signals that remain asserted beyond this limit can be prematurely detected during the next bus cycle.

4.4.3 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. Since the chip-select circuits are driven from the system clock, the bus cycle termination is inherently synchronized with the system clock.

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 $\overline{\text{AS}}$, $\overline{\text{DS}}$, and R/$\overline{\text{W}}$ 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 $\overline{\text{DSACK}}$ option fields in the chip-select option registers determine whether internally generated $\overline{\text{DSACK}}$ or externally generated $\overline{\text{DSACK}}$ are used. Refer to **4.7.1 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. Figure 4–20 shows the $\overline{\text{DSACK}}$ timing for two wait states in read, and a fast-termination read and write.

When fast termination is in use, $\overline{\text{DS}}$ is asserted during read cycles but notduring write cycles. The STRB field in the chip-select option register used must be programmed with the address strobe encoding in order to assert the chip select signal for a fast-termination write.

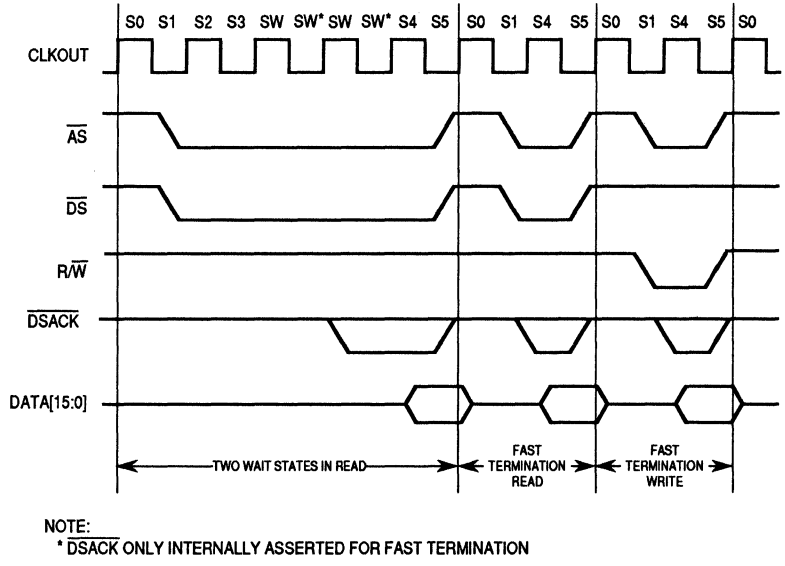


Figure 4–20. Fast-Termination Timing

4.4.3.1 Fast-Termination Read Cycle

A fast-termination read cycle takes place in much the same way as a regular read cycle, except that the clock states for external handshaking are omitted.

State 0 (S0) — The read cycle starts. The MCU places an address on ADDR[23:0] and function codes on FC[2:0]. In the MC68HC16Z1, ADDR[23:20] always follow the state of ADDR19, and FC2 is always equal to one. The MCU drives R/$\overline{W}$ high for a read cycle. Size signals SIZ[1:0] become valid, indicating the number of bytes to be read.

State 1 (S1) — The MCU asserts $\overline{AS}$ indicating that the address on the address bus is valid. The MCU also asserts $\overline{DS}$, indicating to external devices that data can be placed on the data bus. External logic decodes ADDR[23:0], FC[1:0], R/$\overline{W}$, and SIZ[1:0]. One or both of DATA[15:8] and DATA[7:0] are selected, and the responding device places data on that portion of the bus.

State 4 (S4) — Appropriate internal $\overline{DSACK}$ signals are generated and the MCU latches data on the falling edge of S4.

State 5 (S5) — The MCU negates $\overline{AS}$ and $\overline{DS}$, but holds the address valid to provide address hold time for memory systems. R/$\overline{W}$, SIZ[1:0], and FC[2:0] also remain valid throughout S5. The external device must maintain data until it detects the negation of either $\overline{AS}$ or $\overline{DS}$; it must remove the data within approximately one clock period after sensing the negation of $\overline{AS}$ or $\overline{DS}$. Signals that remain asserted longer can be prematurely detected during the next bus cycle.

4.4.3.2 Fast-Termination Write Cycle

A fast-termination write cycle takes place in much the same way as a regular write cycle, except that the clock states for external handshaking are omitted.

State 0 (S0) — The MCU places an address on ADDR[23:0] and function codes on FC[2:0]. In the MC68HC16Z1, ADDR[23:20] always follow the state of ADDR19, and FC2 is always equal to one. The MCU drives R/$\overline{W}$ low for a write cycle. Size signals SIZ[1:0] become valid, indicating the number of bytes to be written.

State 1 (S1) — The MCU asserts $\overline{AS}$, indicating that the address on the address bus is valid. External logic decodes ADDR[23:0], FC[1:0], R/$\overline{W}$, SIZ[1:0], and $\overline{AS}$.

State 4 (S4) — Data driven onto DATA[15:0] becomes valid, and the selected peripheral latches the data. Appropriate internal $\overline{DSACK}$ signals are generated.

State 5 (S5) — The MCU negates $\overline{AS}$, but holds address and data valid to provide address hold time for memory systems. R/$\overline{W}$, SIZ[1:0], and FC[2:0] also remain valid throughout S5.

4.4.4 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 $\overline{AS}$ is asserted. Refer to **4.3.1.7 Function Codes** for more information on codes and encoding.

4

During a CPU space access, ADDR[19:16] are encoded to reflect the type of access being made. Three encodings are used by the MC68HC16Z1, as shown in Figure 4–21. These encodings represent breakpoint acknowledge (Type $0) cycles, low power stop broadcast (Type $3) cycles, and interrupt acknowledge (Type $F) cycles. Type 0 and type 3 cycles are discussed below. Refer to **4.6 Interrupts** for a comprehensive discussion of interrupt acknowledge bus cycles.

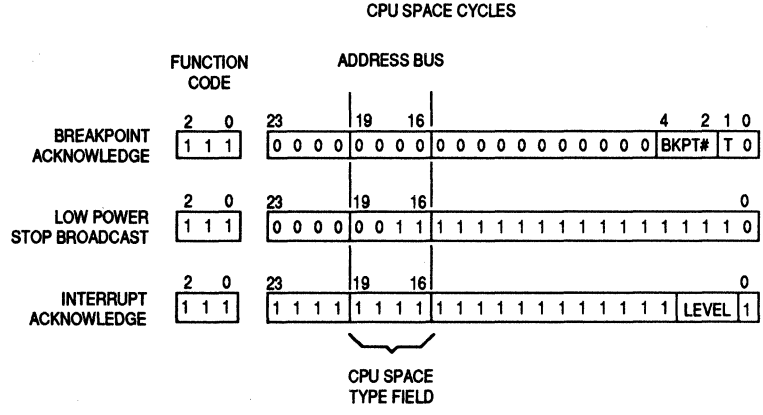


Figure 4–21. CPU Space Address Encoding

4.4.4.1 Breakpoint Acknowledge Cycle

Breakpoints are used to stop program execution at a predefined point during system development. In the MC68HC16Z1, breakpoints are treated as a type of exception processing. Breakpoints can be used alone or in conjunction with the background debugging mode. See **SECTION 5 CENTRAL PROCESSING UNIT** for more information on exception processing and the background debugging mode.

The MC68HC16Z1 has only one source and type of breakpoint — a hardware breakpoint initiated by assertion of the $\overline{\text{BKPT}}$ input. Other modular microcontrollers may have more than one source or type. The breakpoint acknowledge cycle discussed here is the bus cycle that occurs as a part of breakpoint exception processing when a breakpoint is initiated while background debugging mode is not enabled.

$\overline{\text{BKPT}}$ is sampled on the same clock phase as data. If $\overline{\text{BKPT}}$ is valid, the data is tagged as it enters the CPU pipeline. When $\overline{\text{BKPT}}$ is asserted while data is valid during an instruction prefetch, the acknowledge cycle occurs immediately after that instruction has executed. When $\overline{\text{BKPT}}$ is asserted while data is valid during an operand fetch, the acknowledge cycle occurs immediately after execution of the instruction during which it is latched. If $\overline{\text{BKPT}}$ is asserted for only one bus cycle and a pipe flush occurs before $\overline{\text{BKPT}}$ is detected by the CPU, no acknowledge cycle occurs. To ensure detection, $\overline{\text{BKPT}}$ can be asserted until a breakpoint acknowledge cycle is recognized.

When $\overline{\text{BKPT}}$ assertion is acknowledged by the CPU, the MC68HC16Z1 performs a word read from CPU space address $00001E. This corresponds to the breakpoint number field (ADDR[4:2]) and the type bit (T) being set to all ones (source 7, type 1). If this bus cycle is terminated by $\overline{\text{BERR}}$ or by $\overline{\text{DSACK}}$, the MCU performs breakpoint exception processing.

The breakpoint operation flow is shown in Figure 4–22. Figure 4–23 is the timing diagram for the breakpoint acknowledge cycle.

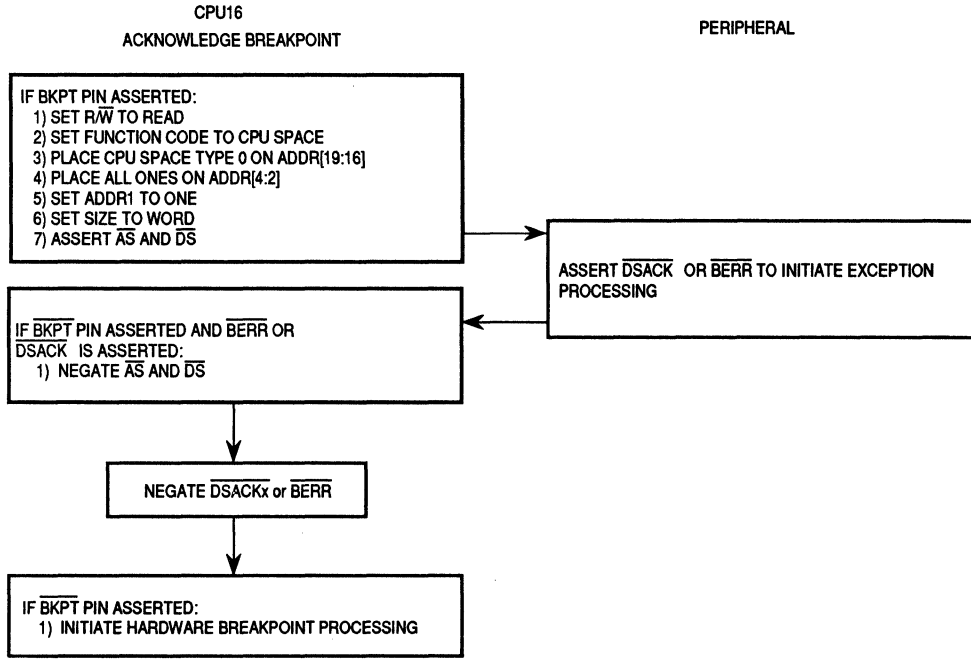


Figure 4–22. Breakpoint Operation Flow

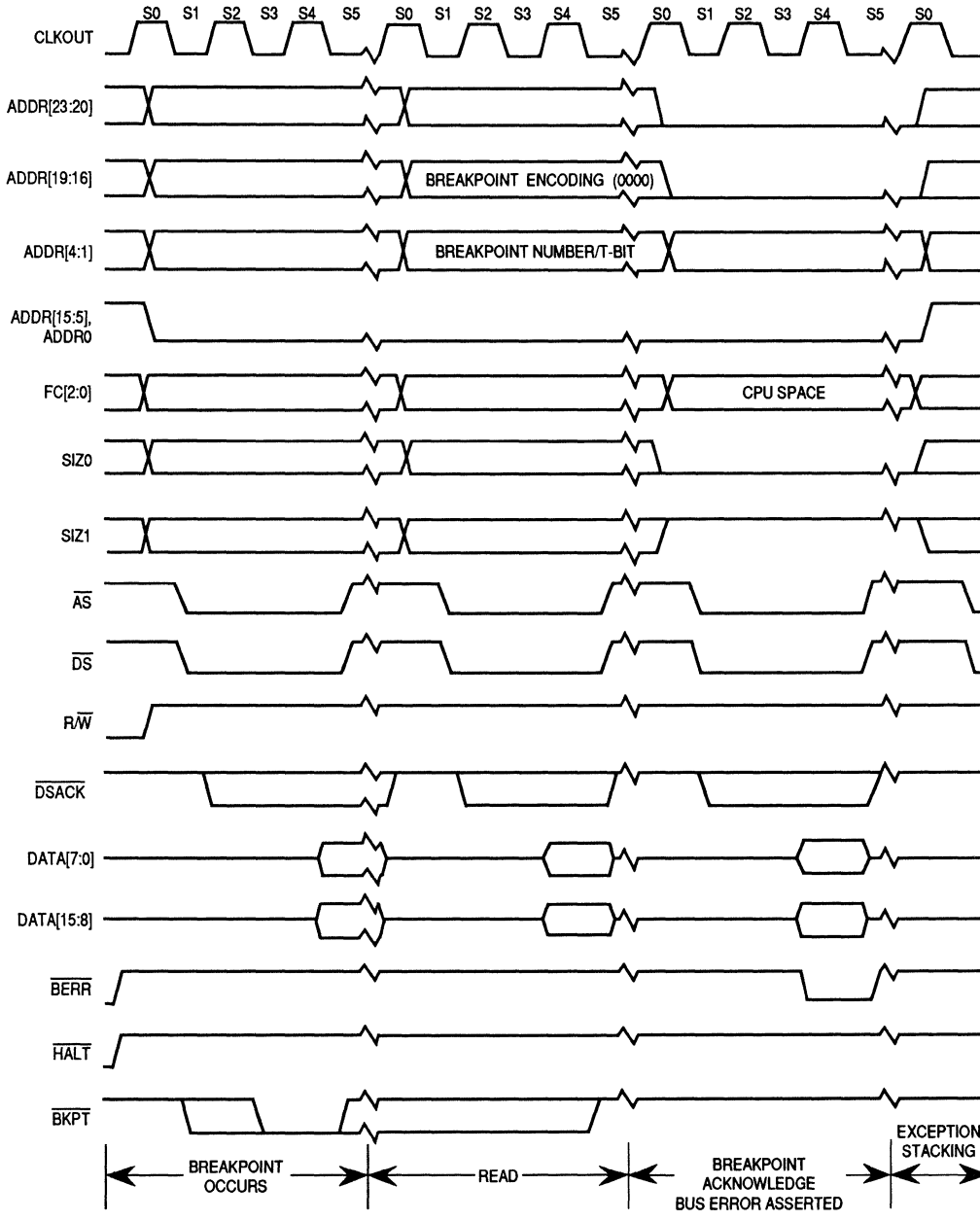


Figure 4–23. Breakpoint Acknowledge Cycle Timing (Exception Signaled)

4.4.4.2 LPSTOP Broadcast Cycle

Low-power stop is initiated by the CPU16. 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 **4.2.4 Low-Power Stop Operation** 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 4–24. 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 4–24. LPSTOP Interrupt Mask Level

4.4.5 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 (refer to **4.3.1 Bus Signals**). Bus exception control cycles are used when bus cycles are not terminated in the expected manner. There are two sources of bus exception control cycles.

Bus error signal ($\overline{\text{BERR}}$).

The internal bus monitor asserts internal $\overline{\text{BERR}}$ when neither $\overline{\text{DSACK}}$ nor $\overline{\text{AVEC}}$ is asserted within a specified period after assertion of $\overline{\text{AS}}$.

The spurious interrupt monitor asserts internal $\overline{\text{BERR}}$ when an interrupt request is acknowledged and no IARB contention occurs. $\overline{\text{BERR}}$ assertion terminates a cycle and causes the MCU to process a bus error exception.

External devices can assert $\overline{\text{BERR}}$ to indicate an external bus error.

Halt signal (HALT).

HALT can be asserted by an external device to cause single bus cycle operation. HALT is typically used for debugging purposes.

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

The acceptable bus cycle terminations for asynchronous cycles in relation to DSACKx assertion are summarized in Table 4–14.

Table 4–14. DSACK, BERR, and HALT Assertion Results

Type of Termination	Control Signal	Asserted on Rising Edge of State		Description of Result
		S	S+2	
NORMAL	DSACK	A	RA	Normal cycle terminate and continue.
	BERR	NA	NA	
	HALT	NA	X	
HALT	DSACK	A	RA	Normal cycle terminate and halt.
	BERR	NA	NA	Continue when HALT is negated.
	HALT	A/RA	RA	
BUS ERROR 1	DSACK	NA/A	X	Terminate and take bus error exception.
	BERR	A	RA	
	HALT	NA	X	
BUS ERROR 2	DSACK	A	X	Terminate and take bus error exception.
	BERR	A	RA	
	HALT	NA	NA	
BUS ERROR 3	DSACK	NA/A	X	Terminate and take bus error exception.
	BERR	A	RA	
	HALT	A/S	RA	
BUS ERROR 4	DSACK	A	X	Terminate and take bus error exception.
	BERR	NA	A	
	HALT	NA	A	

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

4.4.5.1 Bus Errors

The CPU16 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.

$\overline{\text{BERR}}$ assertions do not force immediate exception processing. The signal is synchronized with normal bus cycles and is latched into the CPU16 at the end of the bus cycle in which it was asserted. Since 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:

4

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 CPU16 instruction register, with indeterminate results.

4.4.5.2 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 concerning exceptions). However, two special cases of bus error, called double bus faults, 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 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.

Multiple bus errors within a single instruction which can generate multiple bus cycles cause a single bus error exception after the instruction has 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 **4.4.6 Bus Arbitration**). A bus error or address error that occurs after exception processing has completed (during the execution of the exception handler routine, or later) does not cause a double bus fault.

4.4.5.3 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 (refer to Figure 4–25).

Negating and reasserting $\overline{\text{HALT}}$ in accordance with 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 which 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 CPU16 to process a bus error exception.

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 **4.4.6 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.

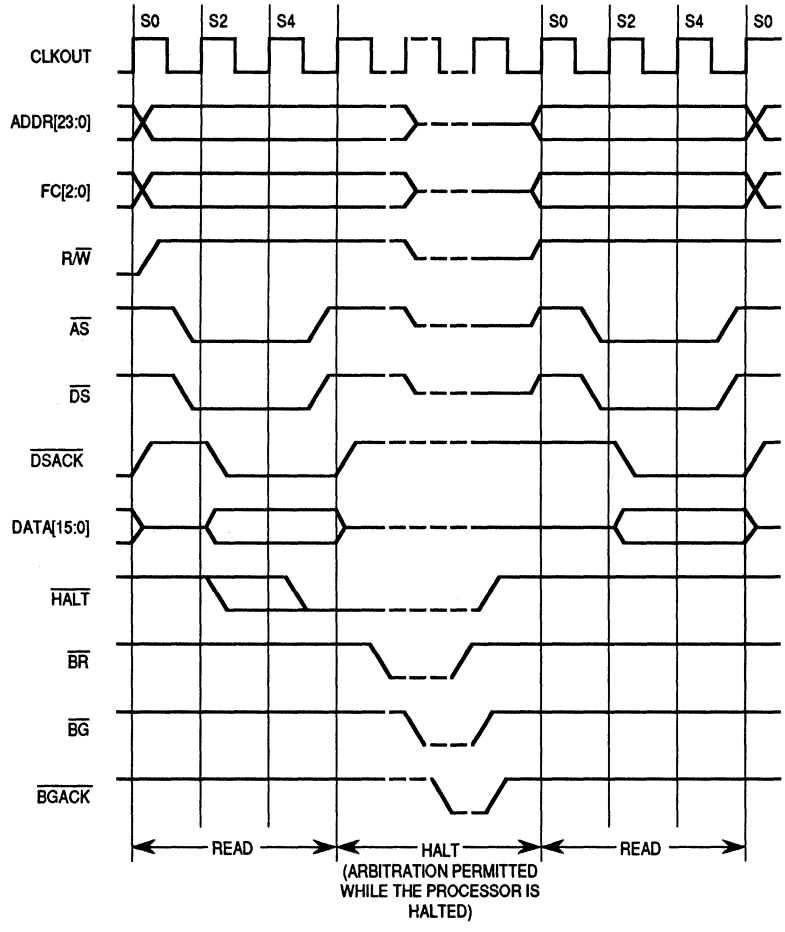


Figure 4–25. HALT Timing

4.4.6 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, 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 ($\overline{BR}$);

B. The MCU asserts the bus grant signal ($\overline{BG}$) to indicate that the bus is available;

C. An external device asserts the bus grant acknowledge ($\overline{BGACK}$) signal to indicate that it has assumed bus mastership.

$\overline{BR}$ can be asserted during a bus cycle or between cycles. $\overline{BG}$ is asserted in response to $\overline{BR}$. To guarantee operand coherency, $\overline{BG}$ is only asserted at the end of operand transfer.

If more than one external device can be bus master, required external arbitration must begin when a requesting device receives $\overline{BG}$. An external device must assert $\overline{BGACK}$ when it assumes mastership, and must maintain $\overline{BGACK}$ 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 $\overline{BG}$ through the arbitration process, and $\overline{BGACK}$ must be inactive, indicating that no other bus master is active. This technique allows processing of bus requests during data transfer cycles.

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

Figure 4–26 is a flowchart of bus arbitration for a single device. The flowchart shows $\overline{BR}$ negated at the same time $\overline{BGACK}$ is asserted.

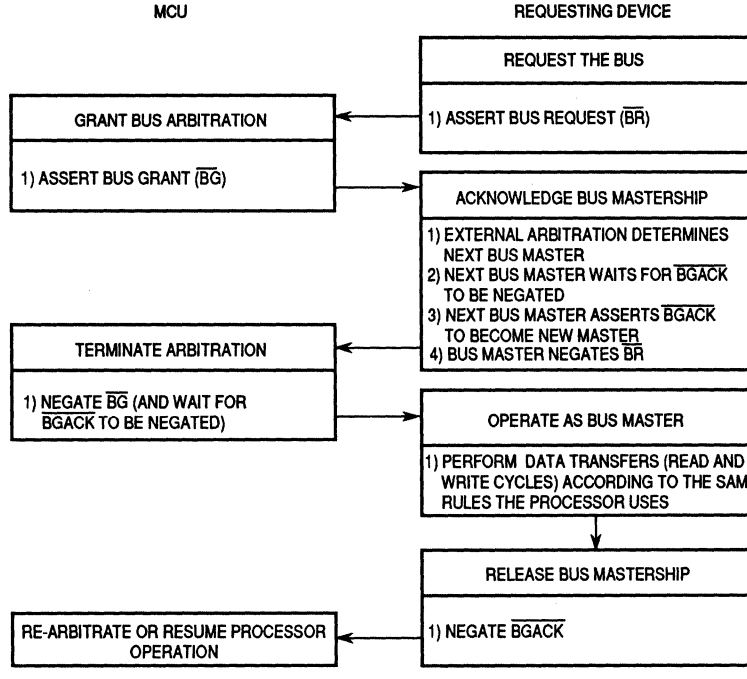


Figure 4–26. Bus Arbitration Flowchart for Single Request

4.4.6.1 Bus Request

External devices capable of becoming bus masters request the bus by asserting the $\overline{BR}$ signal. In a system with a number of devices capable of bus mastership, a wired-OR can connect the bus request line from each device to the MCU. After it has completed any current bus cycle, the MCU asserts $\overline{BG}$, then releases the bus when $\overline{BGACK}$ is asserted.

If no acknowledge signal is received, the MCU remains bus master. This prevents interference with ordinary processing if the arbitration circuitry responds to noise or if an external device negates a request after mastership has been granted.

4.4.6.2 Bus Grant

The MC68HC16Z1 supports operand coherency. When an operand transfer requires multiple bus cycles, the MCU does not release the bus until the entire transfer is complete. The assertion of bus grant is subject to certain constraints:

The minimum time for $\overline{BG}$ assertion after $\overline{BR}$ assertion depends on internal synchronization.

During an external transfer, the MCU does not assert $\overline{BG}$ until after the last cycle of the transfer (determined by SIZ[1:0] and $\overline{DSACK}$[1:0] signals).

When SHEN bits are both set and the CPU is making internal accesses, the MCU does not assert $\overline{BG}$ until the CPU finishes the internal transfers.

Externally, the $\overline{BG}$ signal can be routed through a daisy-chained network or a priority-encoded network. The MCU is not affected by the method of arbitration as long as the protocol is obeyed.

4.4.6.3 Bus Grant Acknowledge

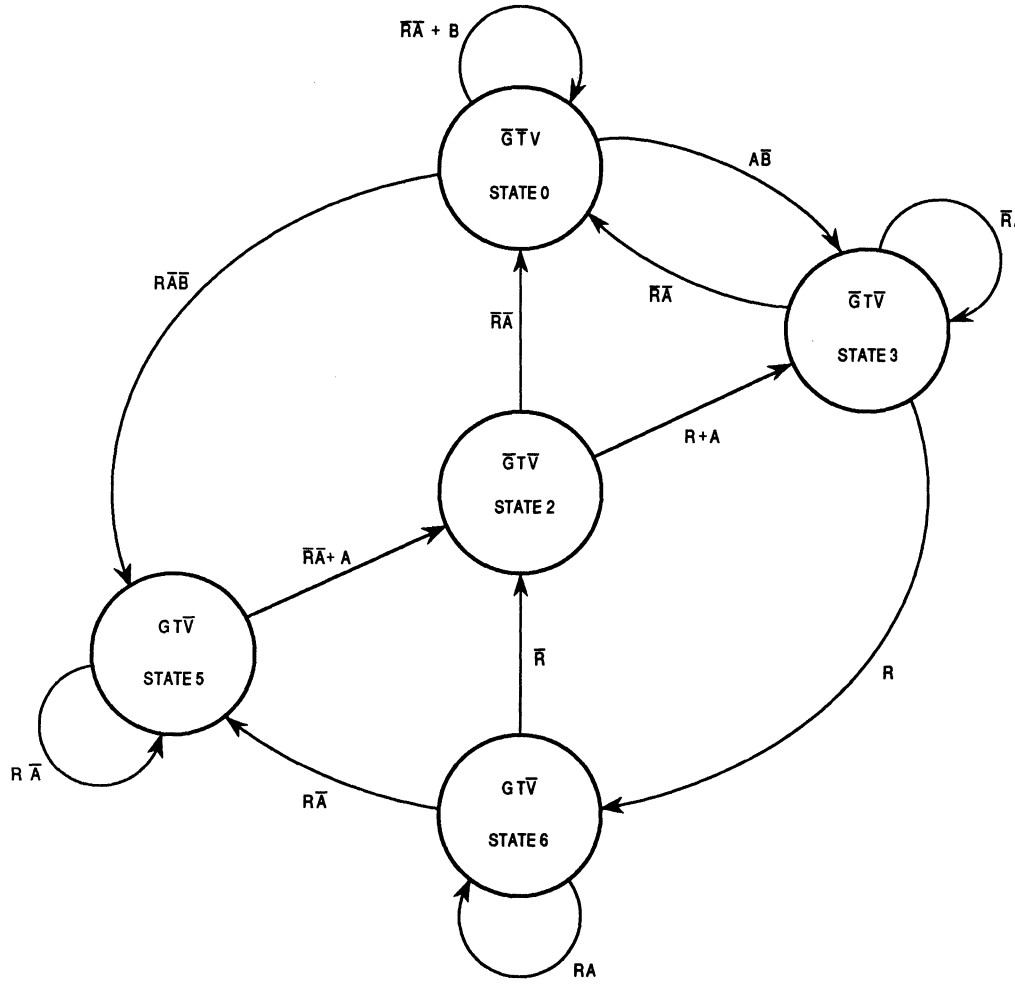
When bus protocols are obeyed, a device becomes the active bus master when it asserts $\overline{BGACK}$. An external device cannot request and be granted the bus while another device is the active bus master. A device remains the bus master until it negates $\overline{BGACK}$. $\overline{BGACK}$ must not be negated until all required bus cycles are completed.

When a device receives the bus and asserts $\overline{BGACK}$, it must also negate $\overline{BR}$. If $\overline{BR}$ remains asserted after $\overline{BGACK}$ assertion, the MCU assumes that another device is requesting the bus and prepares to issue another $\overline{BG}$.

Since external devices have priority, the MCU cannot regain control of the external bus until all pending external bus requests have been satisfied.

4.4.6.4 Bus Arbitration Control

The bus arbitration control unit in the MCU is implemented with a finite-state machine. All asynchronous inputs to the MCU are internally synchronized in a maximum of two CLKOUT cycles. Figure 4–27 is the bus arbitration state diagram. Input signals labeled R and A are internal versions of the bus request and bus grant acknowledge signals that are internally synchronized to the system clock. The bus grant output is labeled G and the internal high-impedance control signal is labeled T. If T is true, the address, data, and control buses are placed in the high-impedance state after the next rising edge following the negation of $\overline{AS}$.



R - BUS REQUEST
A - BUS GRANT ACKNOWLEDGE
B - BUS CYCLE IN PROGRESS
G - BUS GRANT
T - THREE-STATE SIGNAL TO BUS CONTROL
V - BUS AVAILABLE TO BUS CONTROL

NOTE:
ALL FIGURES ARE SHOWN IN POSITIVE LOGIC (ACTIVE HIGH) REGARDLESS OF THEIR TRUE ACTIVE STATE.

Figure 4–27. Bus Arbitration State Diagram

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.

4.4.6.5 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.

4.4.6.6 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 **4.1.3 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. Since internal cycles normally continue to run when the external bus is granted away, 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.

A state-by-state description of show cycle timing follows. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for specific timing information.

State 0 (S0) — Address and function codes become valid, R/$\overline{W}$ is driven to indicate a show read or write cycle, and the size pins indicate the number of bytes to transfer. During a read, the addressed peripheral drives the data bus, and the user must take care to avoid bus conflicts.

State 41 (S41) — $\overline{DS}$ is asserted to indicate that address information is valid.

State 42 (S42) — No change. The bus controller remains in S42 until the internal read cycle is complete.

State 43 (S43) — $\overline{DS}$ is negated to indicate that show data is valid on the next falling edge of CLKOUT. External data bus drivers are enabled so that data becomes valid on the external bus as soon as it is available on the internal bus.

State 0 (S0) — ADDR[23:0], FC[2:0], R/$\overline{W}$, and SIZ[1:0] pins change state to begin the next cycle. Data from the preceding cycle is valid through S0.

4.5 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 in order 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 MC68HC16Z1 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 CPU16.

4.5.1 Reset Exception Processing

The CPU16 processes resets as a type of asynchronous exception. An exception is an event that preempts normal processing. 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 a vector table located in the first 512 bytes of address bank 0. The CPU16 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 CPU16 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.

4.5.8 Reset Processing Summary contains details of exception processing.

4.5.2 Reset Control Logic

SIM reset control logic determines the cause of a reset, synchronizes request signals to CLKOUT, and asserts reset control signals. Reset control logic can drive three different internal signals.

EXTRST (external reset) drives the external reset pin.

CLKRST (clock reset) resets the clock module.

MSTRST (master reset) goes to all other internal circuits.

All resets are gated by CLKOUT. Asynchronous resets are assumed to be catastrophic. An asynchronous reset can occur on any clock edge. 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. Table 4–15 is a summary of reset sources.

Table 4–15. Reset Source Summary

Type	Source	Timing	Reset Lines Asserted by Controller		
EXTERNAL	External	Synchronous	MSTRST	CLKRST	EXTRST
POWER UP	EBI	Asynchronous	MSTRST	CLKRST	EXTRST
SOFTWARE WATCHDOG	Monitor	Asynchronous	MSTRST	CLKRST	EXTRST
HALT	Monitor	Asynchronous	MSTRST	CLKRST	EXTRST
LOSS OF CLOCK	Clock	Synchronous	MSTRST	CLKRST	EXTRST
TEST	Test	Synchronous	MSTRST	—	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 4–28.

4.5.3 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	$\overline{\text{CS6}}$ $\overline{\text{CS7}}$ – $\overline{\text{CS6}}$ $\overline{\text{CS8}}$ – $\overline{\text{CS6}}$ $\overline{\text{CS9}}$ – $\overline{\text{CS6}}$ $\overline{\text{CS10}}$ – $\overline{\text{CS6}}$	ADDR19 ADDR[20:19] ADDR[21:19] ADDR[22:19] ADDR[23:19]
DATA8	$\overline{\text{DSACK0}}$, $\overline{\text{DSACK1}}$, $\overline{\text{AVEC}}$, $\overline{\text{DS}}$, $\overline{\text{AS}}$, SIZE	PORTE
DATA9	$\overline{\text{IRQ7}}$ – $\overline{\text{IRQ1}}$ 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

4.5.3.1 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 prior to the first bus cycle after reset in order to prevent conflict with external memory devices. The first bus cycle occurs 10 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 4–28 shows a recommended method for conditioning the mode select signals.

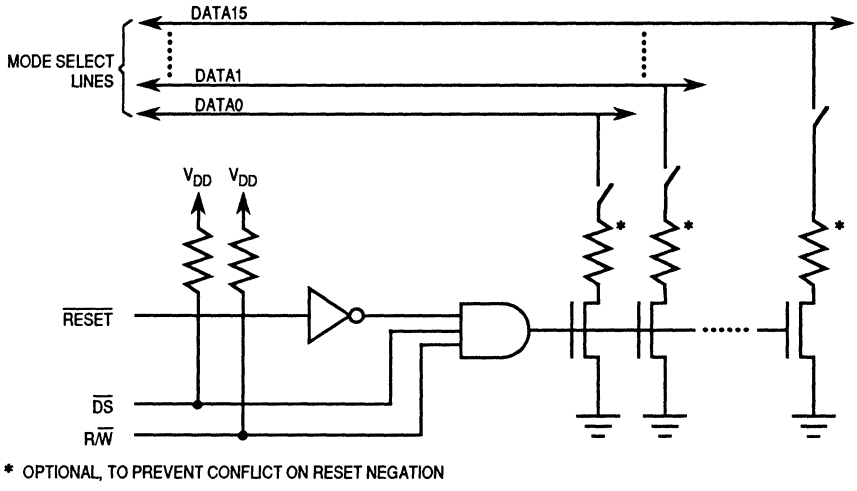


Figure 4–28. Data Bus Mode Select Conditioning

Data bus mode select current is specified in **APPENDIX A ELECTRICAL CHARACTERISTICS**. Do not confuse pin function with pin electrical state. Refer to **4.5.5 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 from word addresses $0000 to $0006 in bank 0 of 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 8 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 **4.7.4 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, CS[8:6] are assigned alternate function as ADDR[21:19], and CS[10:9] remain chip-selects. Because ADDR[23:20] follow the state of ADDR19 in the CPU16, DATA[7:4] have limited use. Refer to **4.7.4 Chip-Select Reset Operation** for more information.

DATA8 determines the function of the $\overline{\text{DSACK0}}$, $\overline{\text{DSACK1}}$, $\overline{\text{AVEC}}$, $\overline{\text{DS}}$, $\overline{\text{AS}}$, and SIZE pins. If DATA8 is held low during reset, these pins are used for discrete 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 used for discrete 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 MC68HC16Z1.

4.5.3.2 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 **4.2 System Clock** for more information.

NOTE

If the MODCLK pin is also used as a parallel port pin, make certain that bus loading does not overcome the weak internal pull-up driver during reset and cause inadvertent clock mode selection.

4.5.3.3 Breakpoint Mode Selection

The MC68HC16Z1 uses internal and external breakpoint ($\overline{\text{BKPT}}$) signals. During reset exception processing, at the release of the $\overline{\text{RESET}}$ signal, the CPU16 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 CPU16 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, BKPT 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 **4.4.4 CPU Space Cycles** for information concerning breakpoint acknowledge bus cycles.

4.5.4 MCU Module Pin Function During Reset

Generally, 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
ADC	PADA[7:0]/AN[7:0]	DISCRETE INPUT
	V_{RH}	REFERENCE VOLTAGE
	V_{RL}	REFERENCE VOLTAGE
CPU	DSI/IPIPE1	DSI/IPIPE1
	DSO/IPIPE0	DSO/IPIPE0
	$\overline{\text{BKPT}}$/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

4.5.5 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 **4.5.7 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

4.5.5.1 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	Pin State While RESET Asserted	Pin State After RESET Released	
		Default Function	Alternate Function
ADDR23/CS10	1	1	ADDR23
ADDR[22:19]/CS[9:6]/PC[6:3]	1	1	ADDR[22:19]
ADDR[18:0]	INACTIVE	ADDR[18:0]	ADDR[18:0]
AS/PE5	INACTIVE	OUTPUT	INPUT
AVEC/PE2	INACTIVE	INPUT	INPUT
BERR	INACTIVE	INPUT	INPUT
BG/CS1	1	1	1
BGACK/CS2	1	1	INPUT
BR/CS0	1	1	INPUT
CLKOUT	OUTPUT	OUTPUT	OUTPUT
CSBOOT	INACTIVE	0	0
DATA[15:0]	MODE SELECT	INPUTS	INPUTS
DS/PE4	INACTIVE	OUTPUT	INPUT
DSACK0/PE0	INACTIVE	INPUT	INPUT
DSACK1/PE1	INACTIVE	INPUT	INPUT
FC[2:0]/CS[5:3]/PC[2:0]	1	1	FC[2:0]
HALT	INACTIVE	INPUT	INPUT
IRQ[7:1]/PF[7:1]	INACTIVE	INPUT	INPUT
MODCLK/PF0	MODE SELECT	INPUT	INPUT
R/W	INACTIVE	OUTPUT	R/W
RESET	ASSERTED	INPUT	INPUT
SIZ[1:0]/PE[7:6]	INACTIVE	SIZ[1:0]	INPUT
TSC	MODE SELECT	INPUT	INPUT

4.5.5.2 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. However, during power-up reset, module port pins may be in an indeterminate state for a short period. Refer to **4.5.7 Power-On Reset** for more information. Table 4–19 is a summary of module pin states.

Table 4–19. Module Pin Reset States

Module	Pin Mnemonic	State
ADC	PADA[7:0]/AN[7:0]	INPUT
	V_{RH}	APPLIED VOLTAGE
	V_{RL}	APPLIED VOLTAGE
CPU	DSI/IPIPE1	IPIPE1 SIGNAL
	DSO/IPIPE0	IPIPE0 SIGNAL
	$\overline{\text{BKPT}}$/DSCLK	$\overline{\text{BKPT}}$ SIGNAL
GPT	PGP7/IC4/OC5	INPUT
	PGP[6:3]/OC[4:1]	INPUT
	PGP[2:0]/IC[3:1]	INPUT
	PAI	INPUT
	PCLK	INPUT
	PWMA, PWMB	INPUT
QSM	PQS7/TXD	INPUT
	PQS[6:4]/PCS[3:1]	INPUT
	PQS3/PCS0/$\overline{\text{SS}}$	INPUT
	PQS2/SCK	INPUT
	PQS1/MOSI	INPUT
	PQS0/MISO	INPUT
	RXD	RXD SIGNAL

4.5.6 Reset Timing

The $\overline{\text{RESET}}$ input must be asserted for a specified minimum period in order for reset to occur (See **APPENDIX A ELECTRICAL CHARACTERISTICS**, Table A–5). External $\overline{\text{RESET}}$ assertion can be delayed internally for a period equal to the longest bus cycle time (or the bus monitor timeout period) in order to protect write cycles from being aborted by reset. While $\overline{\text{RESET}}$ is asserted, SIM pins are either in an inactive, high-impedance state or are driven to their inactive states.

When an external device asserts $\overline{\text{RESET}}$ for the proper period, reset control logic clocks the signal into an internal latch. The control logic drives the $\overline{\text{RESET}}$ pin low for an additional 512 CLKOUT cycles after it detects that the $\overline{\text{RESET}}$ 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 $\overline{\text{RESET}}$ pin goes to an inactive, high-impedance state for 10 cycles. At the end of this 10-cycle period, pin state is tested. When pin state is logic level one, reset exception processing begins. If, however, pin state is logic level zero, 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 10 cycles, and then is tested again. The process repeats until $\overline{\text{RESET}}$ goes high.

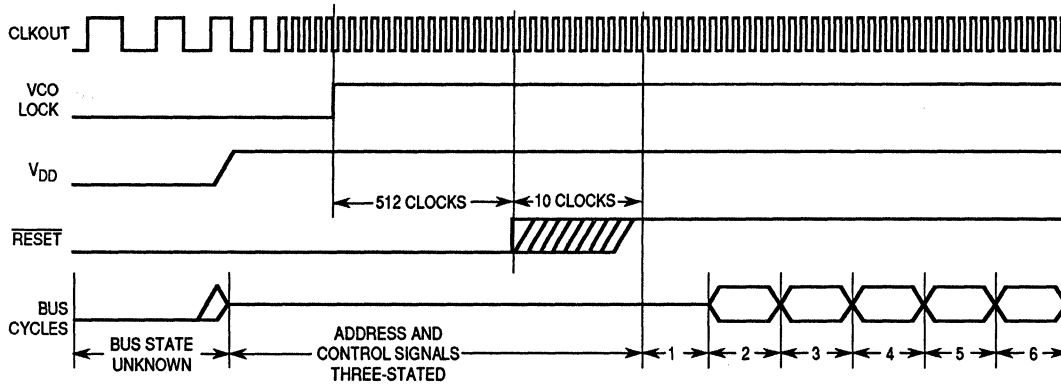
4.5.7 Power-On Reset

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} in order for the MCU to operate. The following discussion assumes that V_{DDSYN} is applied before and during reset — this 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 4-18. 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 means of 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 4–29 is a timing diagram of power-up reset. It shows the relationships between $\overline{\text{RESET}}$, V_{DD}, and bus signals.



NOTES:
1. INTERNAL START-UP TIME
2. INITIAL ZK, SK, PK FETCHED
3. INITIAL PC FETCHED
4. INITIAL SP FETCHED
5. INITIAL IZ FETCHED
6. FIRST INSTRUCTION FETCHED

Figure 4–29. Power-On Reset Timing

4.5.7.1 Use of Three-State Control Pin

Asserting the three-state control (TSC) input causes the MCU to put all output drivers in an inactive, high-impedance state. TSC must remain asserted for 10 clock cycles in order for drivers to change state. There are certain constraints on use of TSC during power-up reset:

When the internal clock synthesizer is used (MODCLK held high during reset), synthesizer ramp-up time affects how long the 10 cycles take. Worst case is approximately 20 milliseconds from TSC assertion.

When an external clock signal is applied (MODCLK held low during reset), pins go to high-impedance state as soon after TSC assertion as 10 clock pulses have been applied to the EXTAL pin.

NOTE

When TSC assertion takes effect, internal signals are forced to values that can cause inadvertent mode selection. Once the output drivers change state, the MCU must be powered down and restarted before normal operation can resume.

4.5.8 Reset Processing Summary

In order 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 condition code register is initialized.

1. The IP field is set to $7, disabling all interrupts below priority 7.
2. The S bit is set, disabling LPSTOP mode.
3. The SM bit is cleared, disabling MAC saturation mode.

C. The K register is cleared.

It is important to be aware that all CCR bits that are not initialized are not affected by reset. However, out of power-on reset, these bits will be indeterminate.

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

A. The CPU16 samples the $\overline{\text{BKPT}}$ input.

B. The CPU16 fetches RESET vectors in the following order:

1. Initial ZK, SK, and PK extension field values.
2. Initial PC.
3. Initial SP.
4. Initial IZ value.

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

C. The CPU16 begins fetching instructions pointed to by the initial PK : PC.

4.5.9 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**.

4.6 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 **4.7 Chip Selects** for more information.

4.6.1 Interrupt Exception Processing

The CPU16 handles interrupts as a type of asynchronous exception. An exception is an event that preempts normal processing. 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 a vector table located in the first 512 bytes of address bank 0. The CPU16 uses vector numbers to calculate displacement into the table. Refer to **SECTION 5 CENTRAL PROCESSING UNIT** for more information concerning exceptions.

4.6.2 Interrupt Priority and Recognition

The CPU16 provides for eight levels of interrupt priority (0–7), seven automatic interrupt vectors, and 200 assignable interrupt vectors. All interrupts with priorities less than 7 can be masked by the interrupt priority (IP) field in the condition code register.

Interrupt recognition is based on the states of interrupt request signals $\overline{\text{IRQ}}$[7:1] and the IP mask value. Each of the signals corresponds to an interrupt priority. $\overline{\text{IRQ1}}$ has the lowest priority, and $\overline{\text{IRQ7}}$ has the highest priority.

The IP field consists of three bits (CCR[7:5]). 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 (except for $\overline{\text{IRQ7}}$) from being recognized and processed. When IP contains %000, no interrupt is masked. During exception processing, the IP field is set to the priority of the interrupt being serviced.

Interrupt request signals can be asserted by external devices or by microcontroller modules. Request lines are connected internally by means of a wired-NOR — simultaneous requests of differing priority can be made. Internal assertion of an interrupt request signal does not affect the logic state of the corresponding MCU pin.

External interrupt requests are routed to the CPU16 via the external bus interface and SIM interrupt control logic — the CPU treats external interrupt requests as though they come from the SIM.

External $\overline{IRQ[6:1]}$ are active-low level-sensitive inputs. External $\overline{IRQ7}$ is an active-low transition-sensitive input — it requires both an edge and a voltage level for validity.

$\overline{IRQ[6:1]}$ are maskable. $\overline{IRQ7}$ is nonmaskable. The $\overline{IRQ7}$ input is transition-sensitive in order to prevent redundant servicing and stack overflow. A nonmaskable interrupt is generated each time $\overline{IRQ7}$ is asserted, and each time the priority mask changes from %111 to a lower number while $\overline{IRQ7}$ is asserted.

Interrupt request signals 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 CPU16 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 of equal or lower priority than the current IP mask value is made, the CPU does not recognize the occurrence of the request in any way.

4.6.3 Interrupt Acknowledge and Arbitration

Interrupt acknowledge bus cycles are generated during exception processing. When the CPU16 detects one or more interrupt requests of a priority higher than the interrupt priority mask value, it performs a CPU space read from address $FFFFF : [IP] : 1. (Refer to **4.3.1.7 Function Codes** and **4.4.4 CPU Space Cycles** for more information.)

The CPU space read cycle performs two functions: it places a mask value corresponding to the highest priority interrupt request on the address bus, and it acquires an exception vector number from the interrupt source. The mask value also serves two purposes: it is decoded by modules that have requested interrupt service to determine whether the current interrupt acknowledge cycle pertains to them, and it is latched into the CCR IP field in order to mask lower-priority interrupts during exception processing.

Modules that have requested interrupt service decode the IP value placed on the address bus at the beginning of the interrupt acknowledge cycle, and if their requests are at the specified IP level, respond to the cycle. Arbitration between simultaneous requests of the same priority is performed by means of serial contention between module interrupt arbitration (IARB) field bit values.

Each module that can make an interrupt service request, including the SIM, has an IARB field in its configuration register. An IARB field can be assigned a value from %0001 (lowest priority) to %1111 (highest priority). A value of %0000 in an IARB field causes the CPU16 to process a spurious interrupt exception when an interrupt from that module is recognized.

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. Initialization software must assign different IARB values in order to implement an arbitration scheme.

NOTE

Do not assign the same arbitration priority to more than one module. When two or more IARB fields have the same nonzero value, the CPU16 interprets multiple vector numbers simultaneously, with unpredictable consequences.

Arbitration must always take place, even when a single source is requesting service. This point is important for two reasons: the EBI does not transfer the CPU interrupt acknowledge 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 dominant module must place an interrupt vector number on the data bus and terminate the bus cycle or assert the autovector ($\overline{\text{AVEC}}$) signal. In the case of an external interrupt request, because the interrupt acknowledge cycle is transferred to the external bus, an external device must decode the mask value and respond with a vector number, then generate data and size acknowledge ($\overline{\text{DSACK}}$) cycle termination signals. 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.

The periodic interrupt timer (PIT) in the SIM can generate internal interrupt requests of specific priority at predetermined intervals. By hardware convention, PIT interrupts are serviced before external interrupt service requests of the same priority. Refer to **4.1.11 Periodic Interrupt Timer** for more information.

4.6.4 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. Processor state is stacked, then the CCR PK extension field is cleared.

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] = %11111111111; ADDR[3:1] = the priority of the interrupt request being acknowledged; and ADDR0 = %1.

   3. Request priority is latched into the CCR IP field from the address bus.

D. Modules or external peripherals that have requested interrupt service decode the priority value in ADDR[3:1]. If request priority is the same as the priority value in the address, IARB contention takes place. When there is no contention, the spurious interrupt monitor asserts $\overline{\text{BERR}}$, and a spurious interrupt exception is processed.

E. After arbitration, the interrupt acknowledge cycle can be completed in one of three ways:

   1. The dominant interrupt source supplies a vector number and $\overline{\text{DSACK}}$ signals appropriate to the access. The CPU16 acquires the vector number.

   2. 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 CPU16 generates an autovector number corresponding to interrupt priority.

   3. The bus monitor asserts $\overline{\text{BERR}}$ and the CPU16 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.

4.6.5 Interrupt Acknowledge Bus Cycles

Interrupt acknowledge bus cycles are CPU space cycles that are generated during exception processing. The following paragraphs describe the various kinds of interrupt acknowledge bus cycles that can be executed as part of interrupt exception processing.

4.6.5.1 External Bus Cycle Terminated by Data and Size Acknowledge Signals

The MCU acknowledges an external interrupt request by performing an external read cycle to obtain the interrupt vector number. The following paragraphs describe the interrupt acknowledge cycle for devices that supply a vector number and appropriate bus cycle termination signals.

Other interrupt sources use the autovector cycle described in **4.6.5.2 External Bus Cycle Terminated by External Autovector Signal**. The interrupt acknowledge cycle is a CPU space read cycle. It differs from the read cycle described in **4.4.2.1 Read Cycle** in the following ways:

A. FC[2:0] are set to %111, the CPU space encoding.

B. ADDR[19:16] (the CPU space type field) are set to %1111, the interrupt acknowledge encoding.

C. ADDR[3:1] are set to the interrupt request level and ADDR0 is set to one.

D. All remaining address bits are set.

E. SIZ[1:0] and R/$\overline{W}$ are driven to indicate a single-byte read cycle.

Interrupting devices must decode ADDR[3:1] to determine which device puts the interrupt vector number on the bus. The responding device must also decode SIZ[1:0] for dynamic bus allocation. Because ADDR0 = 1 during an interrupt acknowledge cycle, transfer case is either an odd byte-to-byte transfer or an odd byte-to-word transfer. The vector number is placed on DATA[15:8] if the device is an 8-bit port, or on DATA[7:0] if it is a 16-bit port. To terminate the cycle, the device must assert an appropriate combination of $\overline{DSACK}$[1:0] signals.

Chip-select logic can be programmed to decode this bus cycle and generate an interrupt acknowledge signal. Refer to **4.7.3 Using Chip-Select Signals for Interrupt Acknowledge** for more information.

Figure 4–30 is a flowchart of the cycle. Figure 4–31 shows cycle timing.

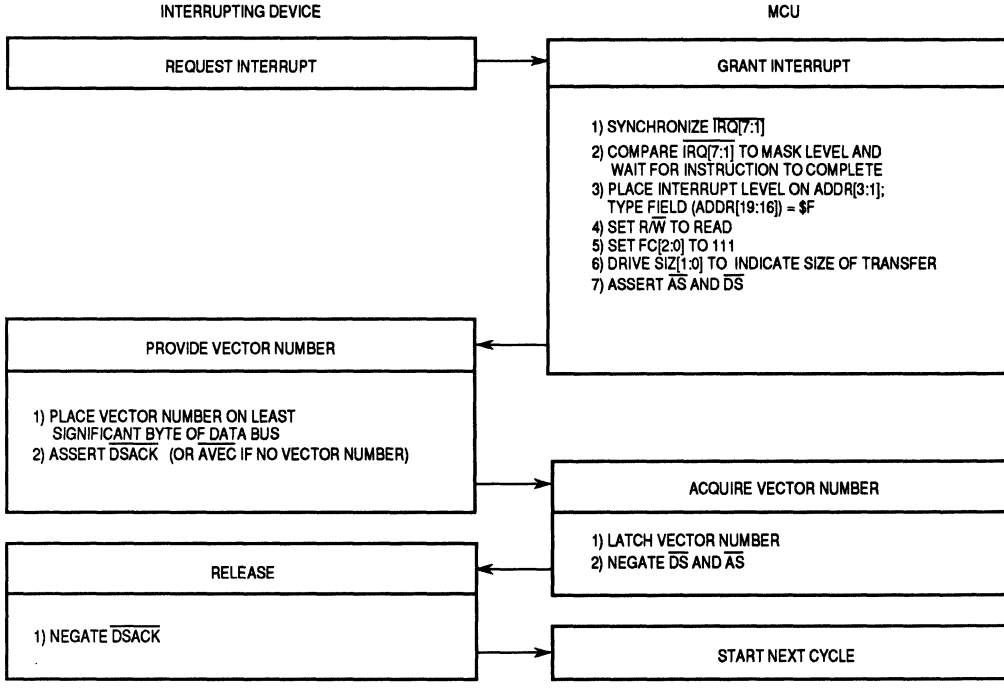


Figure 4–30. Interrupt Acknowledge Cycle Flowchart

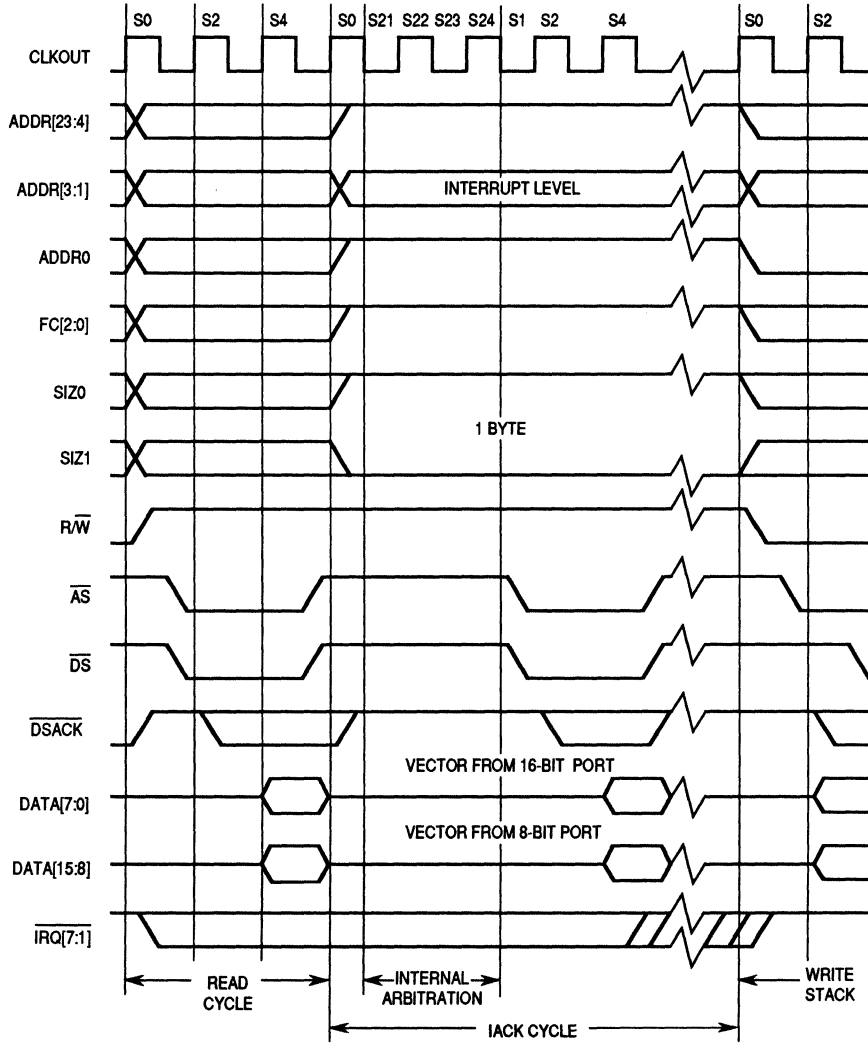


Figure 4–31. Interrupt Acknowledge Cycle Timing

4.6.5.2 External Bus Cycle Terminated by External Autovector Signal

An interrupting device requests an automatically generated vector, or autovector, by asserting the AVEC signal to terminate an interrupt acknowledge cycle. DSACK signals must not be asserted during an interrupt acknowledge cycle terminated by AVEC. If the AVEC pin is wired low, the CPU generates an autovector whenever an interrupt (of any priority, from any source) is acknowledged.

When AVEC is asserted, the CPU ignores the state of the data bus and generates a vector number. The autovector number corresponds to the priority level of the interrupt request. Seven autovectors are available, one for each of the seven interrupt request signals. Figure 4–32 shows the timing for an autovector operation.

Chip-select logic can be programmed to decode this bus cycle and generate an internal AVEC response when an external interrupt request is made. The interrupting device does not have to respond in this case. Chip-select logic is typically used to generate an internal autovector signal when the corresponding chip-select pin is used for an alternate function or for general-purpose I/O. Refer to **4.7.2 Chip-Select Operation** for more information.

4.6.5.3 Spurious Interrupt Cycle

When an interrupt request is made, but no IARB field value is asserted in response to the interrupt acknowledge cycle, the spurious interrupt monitor asserts the BERR signal internally to prevent vector acquisition. When a responding device does not terminate an interrupt acknowledge cycle with AVEC or DSACK, the bus monitor asserts BERR internally. The CPU16 automatically generates the spurious interrupt vector number ($F) in both cases. If the halt signal (HALT) is asserted while internal BERR is asserted, the MCU responds as though BERR alone is asserted.

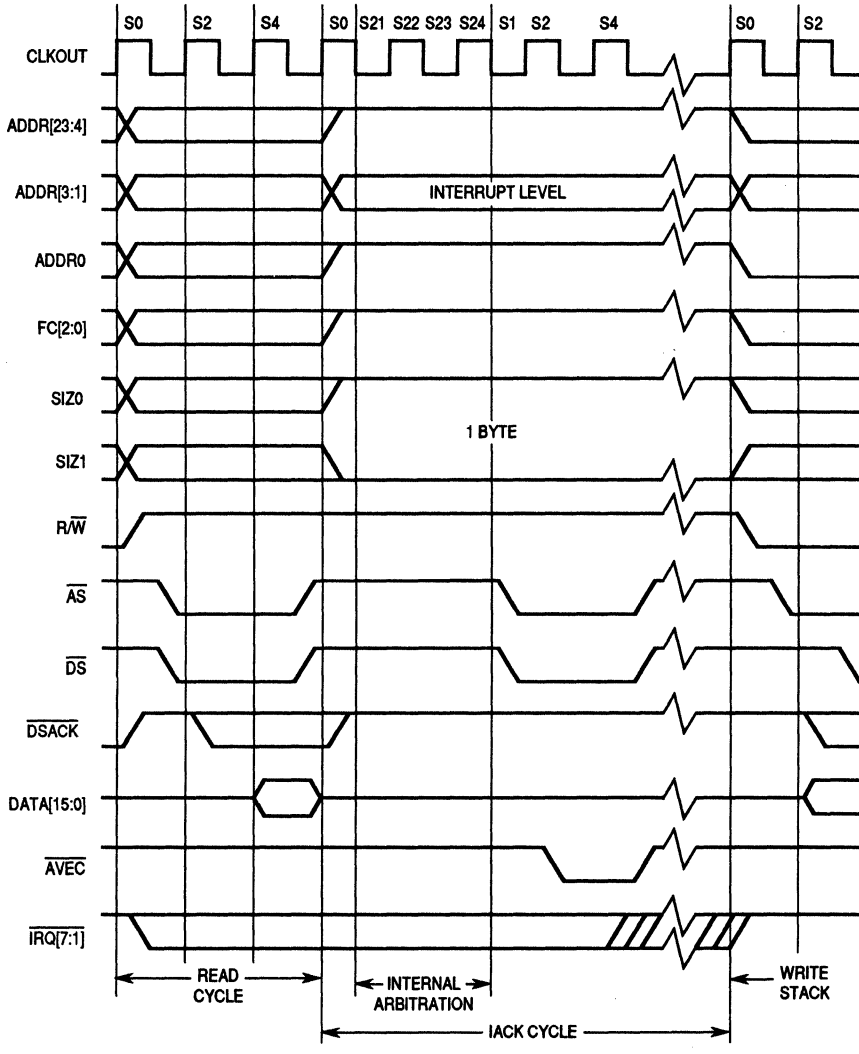
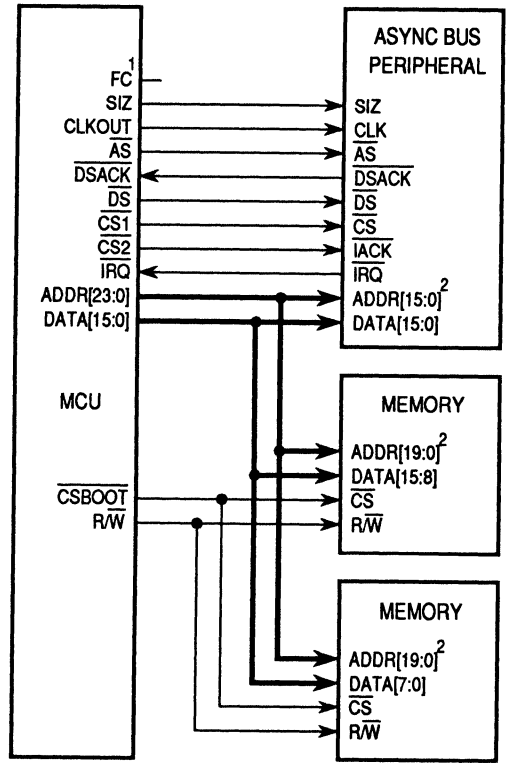


Figure 4–32. Autovector Operation Timing

4.7 Chip Selects

Typical microcontrollers require additional hardware to provide external chip-select signals. The MC68HC16Z1 includes 12 programmable chip-select circuits that can provide two to thirteen cycle access to external memory and

peripherals. Address block sizes of 2 Kbytes to 1 Mbyte can be selected. However, because ADDR[23:20] follow the state of ADDR19, 512-Kbyte blocks are the largest usable size. Figure 4–33 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.

Figure 4–33. Basic M68HC16 System

Chip-select assertion can be synchronized with bus control signals to provide output enable, read/write strobe, or interrupt acknowledge signals. Logic can also generate DSACK and AVEC signals internally. A single DSACK generator is shared by all chip-select circuits — multiple chip selects assigned to the same address must have the same number of wait states. 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, chip-select signals 10 through 0 are disabled, and cannot be asserted until a transfer size is chosen. The boot ROM select signal 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 **4.5.3.1 Data Bus Mode Selection** for more information). Figure 4–34 is a functional diagram of a single chip-select circuit.

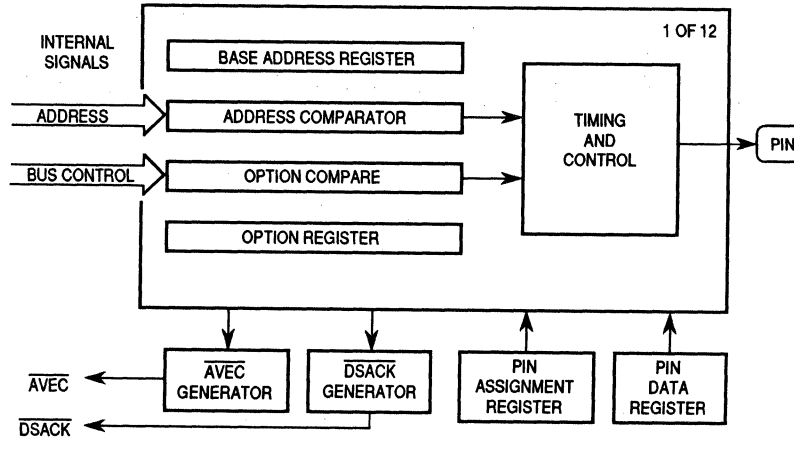


Figure 4–34. Chip-Select Circuit Block Diagram

4.7.1 Chip-Select Registers

Each chip-select pin can have one or more functions. Chip-select pin assignment registers (CSPAR[1:0]) determine functions of the pins. Pin assignment registers also determine port size (8- or 16-bit) for dynamic bus allocation. A pin data register (CSPDR) 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 2 Kbytes to 1 Mbyte can be selected by writing values to the appropriate base address register (CSBAR[10:0], CSBARBT). However, because the logic state of ADDR20 is always the same as the state of ADDR19 in the MC68HC16Z1, the largest usable block size is 512 Kbytes. Address blocks for separate chip-select functions can overlap.

Chip select option registers (CSOR[10:0], 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**.

4.7.1.1 Chip-Select Pin Assignment Registers

The pin assignment registers contain 12 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 4–20.

Table 4–20.
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 4–21 shows pin assignment field encoding. Pins that have no discrete output function do not use the %00 encoding.

Table 4–21.
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 8-bits or 16-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 **4.7.1.3 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 **4.5.3.1 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 ($\overline{\text{CSBOOT}}$) are disabled out of reset. There are 12 chip-select functions and only 8 associated data bus pins — there is not a one-to-one correspondence. Refer to **4.7.4 Chip-Select Reset Operation** for more detailed information.

The $\overline{\text{CSBOOT}}$ signal is normally asserted out of reset. The state of the DATA0 line during reset determines what port width $\overline{\text{CSBOOT}}$ 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.

4.7.1.2 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 4–22 shows BLKSZ encoding.

Table 4–22. 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	512 K	ADDR[23:20]

ADDR[23:20] = ADDR19 during normal operation.

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.

Because the logic state of ADDR[23:20] follows that of ADDR19 in the CPU16, maximum block size is 512 Kbytes. Because ADDR[23:20] follow the logic state of ADDR19, addresses from $080000 to $F7FFFF are inaccessible.

After reset, the MCU fetches initialization values from word addresses $0000 to $0006 in bank 0 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 vectors located at these addresses can be automatically enabled by $\overline{\text{CSBOOT}}$ after a reset. The block size field in CSBARBT has a reset value of 512 Kbyte. Refer to **4.7.4 Chip-Select Reset Operation** for more information.

4.7.1.3 Chip-Select Option Registers

Option register fields determine timing of and conditions for assertion of chip-select signals. Other constraints set by fields in the option register and in the base address register must also be satisfied in order to assert a chip-select signal, and to provide DSACK or autovector support. Table 4–23 is a summary of option register functions.

Table 4–23. Option Register Function Summary

MODE	BYTE	R/W	STRB	DSACK	SPACE	IPL	AVEC
0 = ASYNC*	00 = Disable	00 = Rsvd	0 = AS	0000 = 0 WAIT	00 = CPU SP	000 = All*	0 = Off*
1 = SYNC	01 = Lower	01 = Read	1 = 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 is asynchronous or synchronized to the M6800-type bus clock signal (ECLK) available on ADDR23 (refer to **4.2 System Clock** for more information on ECLK).

Asynchronous chip-select operation corresponds to asynchronous external bus operation. In asynchronous mode, chip-select signal assertion occurs at the same time as AS or DS assertion, depending on the value in the STRB field. R/W determines whether the chip-select signal is asserted for a read only, for a write only, or for both read and write. An asynchronous chip-select cycle must be terminated by a data and size acknowledge (DSACK) signal or by an autovector (AVEC) signal. The DSACK field determines the source of the data and size acknowledge signal and controls wait-state insertion in asynchronous mode. The AVEC bit controls the internally-generated autovector signal. DSACK field encoding %1110 is used to enable fast-termination bus cycles (refer to **4.4.3 Fast Termination Cycles** for more information).

In synchronous mode, chip-select assertion is synchronized to the MCU ECLK output. When a match condition occurs, the chip-select circuit signals the EBI that an ECLK cycle is pending. When the EBI determines that bus timing constraints are satisfied, the chip-select signal is asserted. Transfers of word and long-word data to an 8-bit port are performed consecutively, without insertion of additional ECLK cycles. During synchronous operation, bus monitor timeout period must be longer than the number of clock cycles required for two ECLK cycles (refer to **4.1.7 Bus Monitor** for more information). Because synchronous cycles are not terminated by data and size acknowledge signals, the $\overline{\text{DSACK}}$ field has no effect in synchronous mode. The $\overline{\text{AVEC}}$ bit must not be used in synchronous mode — autovector response timing can vary due to ECLK synchronization.

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 $\overline{\text{CS}}$[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 disable option prevents chip-select signal assertion, even when all other constraints are satisfied. The associated pin is driven high, and associated signals, such as $\overline{\text{DSACK}}$ or $\overline{\text{AVEC}}$, cannot be asserted internally by chip-select logic.

The upper and lower byte options are used to generate chip-select signals for single-byte transfers to 16-bit ports. For example, two chip-select lines can be used to select 8-bit banks in a 16-bit memory. To do this, program two chip-select base address registers with the same base address, then set up the individual lines for byte access. Program both option registers identically except for the BYTE fields — use the upper byte option for one line and the lower byte option for the other.

The both-bytes option is used to generate a single chip-select signal for word transfers to a 16-bit port.

The **R/$\overline{\text{W}}$** 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.

If an internally generated acknowledge option is selected, bus timing can be adjusted by inserting up to 13 wait states before DSACK assertion. A wait state has a duration of one clock cycle. The wait states are inserted beginning with S3 of the external bus cycle. A no-wait-state encoding corresponds to a three-cycle bus.

Fast termination encoding corresponds to a two-cycle bus access. MCU modules typically respond at this rate, but fast termination can also be used to access fast external devices. If an external device is fast enough, the bus cycle can be terminated at S3. (Refer to **4.4.3 Fast Termination Cycles**.)

Cycles are terminated by the first DSACK that occurs — if an external DSACK occurs during internal wait state generation, the bus cycle terminates immediately. If the externally generated acknowledge option is selected, the MCU waits indefinitely for external DSACK assertion.

If multiple chip selects are to be used to provide control signals to a single device and match conditions can occur simultaneously, all but one of the associated DSACK fields should be programmed either for external DSACK or for the same number of wait states. The remaining DSACK field should be programmed for the fast termination option. This prevents a conflict on the internal bus when the wait states are loaded into the DSACK counter shared by all chip selects.

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.

A chip select set up for CPU space access should not be used to select an external device for reading or writing — I/O occurs in supervisor space, but interrupt acknowledge cycles occur in CPU space. A separate chip select is needed to access the external device. The chip select used for the AVEC can, however, still be used for discrete I/O.

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.

If an internally generated acknowledge option is selected, bus timing can be adjusted by inserting up to 13 wait states before DSACK assertion. A wait state has a duration of one clock cycle. The wait states are inserted beginning with S3 of the external bus cycle. A no-wait-state encoding corresponds to a three-cycle bus.

Fast termination encoding corresponds to a two-cycle bus access. MCU modules typically respond at this rate, but fast termination can also be used to access fast external devices. If an external device is fast enough, the bus cycle can be terminated at S3. (Refer to **4.4.3 Fast Termination Cycles**.)

Cycles are terminated by the first DSACK that occurs — if an external DSACK occurs during internal wait state generation, the bus cycle terminates immediately. If the externally generated acknowledge option is selected, the MCU waits indefinitely for external DSACK assertion.

If multiple chip selects are to be used to provide control signals to a single device and match conditions can occur simultaneously, all but one of the associated DSACK fields should be programmed either for external DSACK or for the same number of wait states. The remaining DSACK field should be programmed for the fast termination option. This prevents a conflict on the internal bus when the wait states are loaded into the DSACK counter shared by all chip selects.

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.

A chip select set up for CPU space access should not be used to select an external device for reading or writing — I/O occurs in supervisor space, but interrupt acknowledge cycles occur in CPU space. A separate chip select is needed to access the external device. The chip select used for the AVEC can, however, still be used for discrete I/O.

4.7.2 Chip-Select Operation

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

1. Function codes to SPACE fields.
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 $\overline{DSACK}$ (refer to **4.4.2.1 Read Cycle** and **4.4.2.2 Write Cycle** for more information on wait-state timing). If a peripheral does not generate $\overline{DSACK}$, internal $\overline{DSACK}$ generation must be selected and a predetermined number of wait states may be programmed into the chip-select option register.

Refer to **4.6.5 Interrupt Acknowlege Bus Cycles** for additional information. Figure 4–35 is a flow diagram for the assertion of chip select.

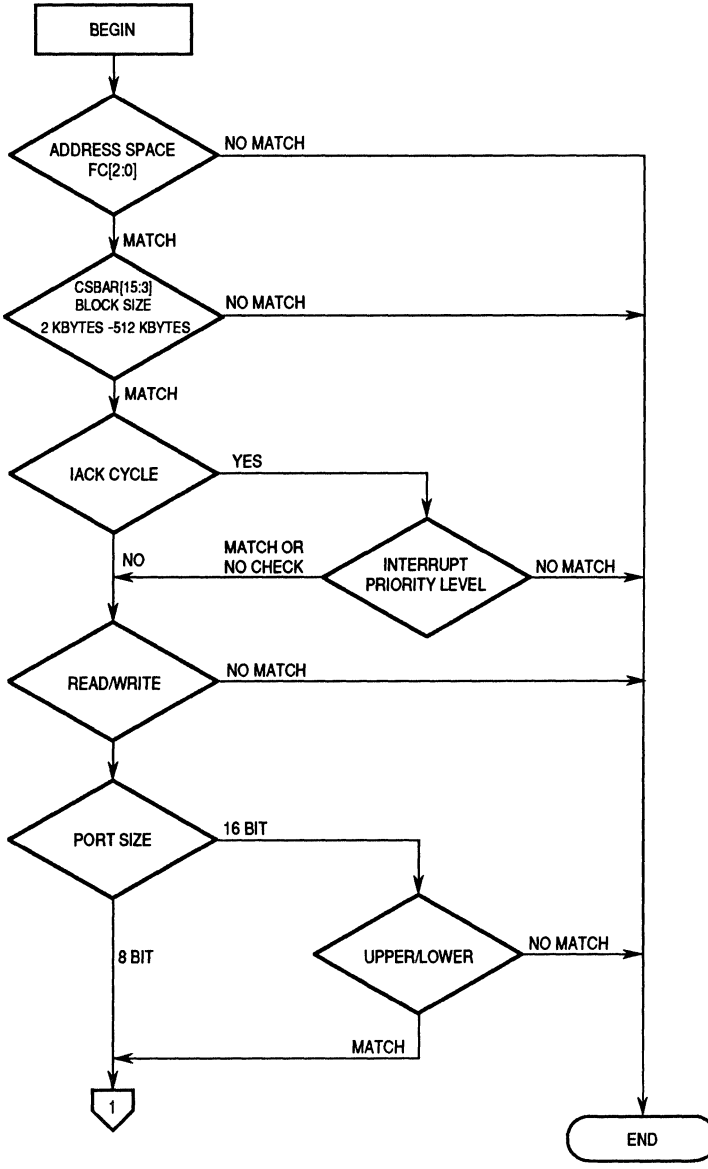


Figure 4–35. Flow Diagram for Chip Select (Sheet 1 of 3)

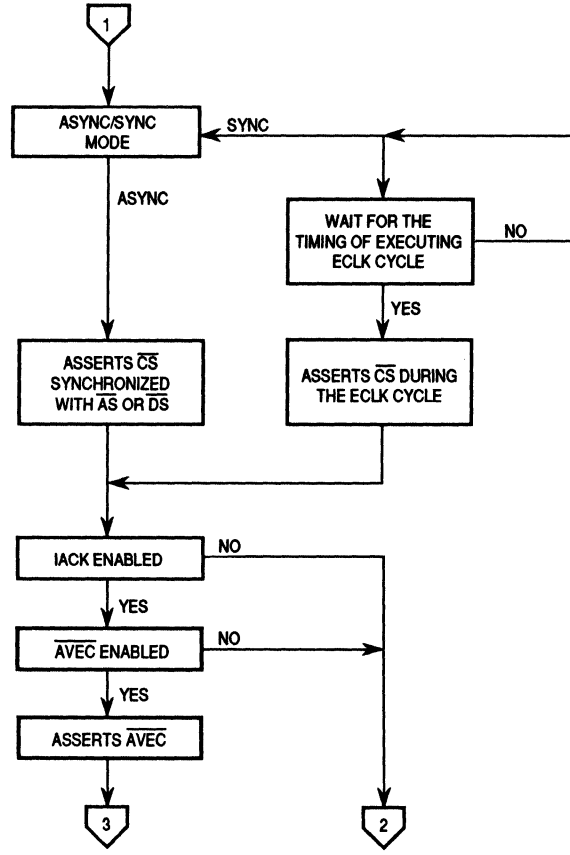


Figure 4–35. Flow Diagram for Chip Select (Sheet 2 of 3)

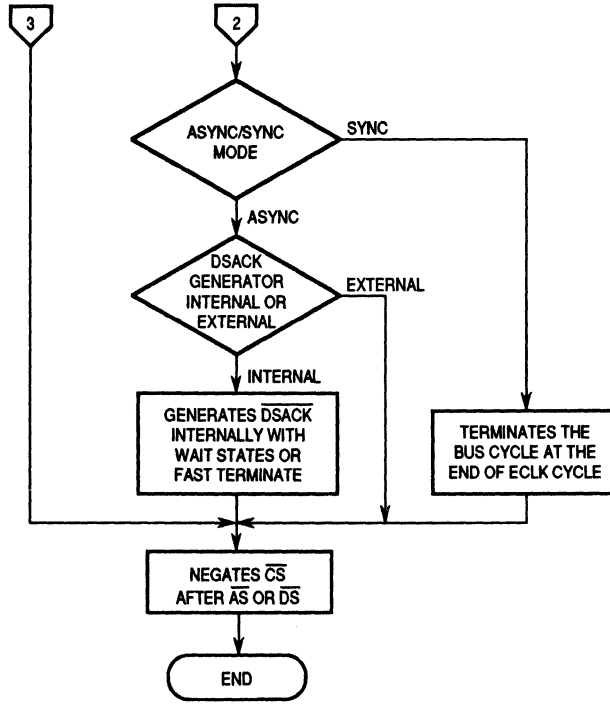


Figure 4–35. Flow Diagram for Chip Select (Sheet 3 of 3)

4.7.3 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 **4.4.4 CPU Space Cycles** and **4.6 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], since the address is compared to an address generated by the CPU. In the MC68HC16Z1, ADDR[23:20] follow the state of ADDR19 — the states of base register bits [15:12] must match that of bit 11.

Figure 4–36 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.

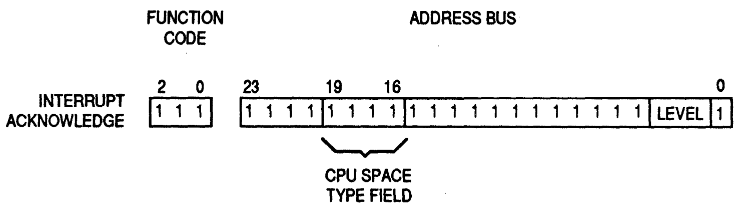


Figure 4–36. CPU Space Encoding for Interrupt Acknowledge

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. Use of CPU space automatically drives ADDR[19:16] to logic level one and ADDR[23:20] follow the state of ADDR19, so the address comparator can use only the lower 16 address lines.
3. Set the R/$\overline{W}$ 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, since 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 — the bus cycle is terminated by asserting AVEC. This can be done either by asserting the AVEC pin or by generating AVEC internally, using the chip-select option register.

4.7.4 Chip-Select Reset Operation

The least significant bits of each of the 2-bit 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, but these drivers can be overcome by bus loading effects.

The CSBOOT assignment field in CSPAR0 is configured differently. The MSB, bit 1 of CSPAR0, has a reset value of one. This enables the CSBOOT signal to select a boot ROM containing initialization firmware. The LSB value, determined by the logic level of DATA0 during reset, selects boot ROM port size. When DATA0 is held low, port size is 8 bits — when DATA0 is held high, port size is 16 bits.

After reset, the MCU fetches initialization values from word addresses $0000 to $0006 in bank 0 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 ROM device containing vectors located at these addresses can be enabled by CSBOOT after a reset. The block size field in CSBARBT has a reset value of 512 Kbyte.

The byte field in option register CSORBT has a reset value of both bytes, but CSOR[10:0] have a reset value of disable, since they should not select external devices until an initial program sets up the base and option registers. Table 4–24 shows the reset values in the base and option registers for CSBOOT.

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

Fields	Reset Values
Base Address	$0000 0000
Block Size	512 Kbyte
Async/Sync Mode	Asynchronous Mode
Upper/Lower Byte	Both Bytes (CSORBT)
Byte	Disable (CSOR10–CSOR0)
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

4.8 Parallel Input/Output Ports

Sixteen of the 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. PE3 is not connected to a pin. PE3 returns zero when read — writes have no effect. Pin assignment registers, data direction registers, and data registers are used to implement discrete I/O.

4.8.1 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 to be a bus control signal. Any bit cleared to zero defines the corresponding pin to be an I/O pin. PEPA3 returns one when read — writes have no effect.

4.8.2 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. DDE3 returns zero when read; writes have no effect.

4.8.3 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.

4.9 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 Motorola use. Register names and addresses are provided in **APPENDIX D REGISTER SUMMARY** to show the user that these addresses are occupied. The $\overline{\text{TSTME}}$ and QUOT pins are also used for factory test.

4

SECTION 5
CENTRAL PROCESSING UNIT

The central processing unit (CPU16) was designed to provide compatibility with the MC68HC11 and to provide additional capabilities associated with 16- and 32-bit data sizes, 20-bit addressing, and digital signal processing. CPU16 registers are an integral part of the CPU and are not addressed as memory locations. The CPU16 register model contains all the resources of the MC68HC11, plus additional resources.

The CPU16 treats all peripheral, I/O, and memory locations as parts of a pseudolinear 1 Megabyte address space. There are no special instructions for I/O that are separate from instructions for addressing memory. Address space is made up of 16 64-Kbyte banks. Specialized bank addressing techniques and support registers provide transparent access across bank boundaries.

The CPU16 interacts with external devices and with other modules within the microcontroller via a standardized bus and bus interface. There are bus protocols for memory and peripheral accesses, as well as for managing an hierarchy of interrupt priorities.

This section is intended to provide an overview of and ready reference to CPU16 function. For detailed information concerning CPU operation, refer to the *CPU16 Reference Manual* (CPU16RM/AD).

5.1 Register Model

Figure 5–1 shows the CPU16 register model. Registers are discussed in detail in the following paragraphs.

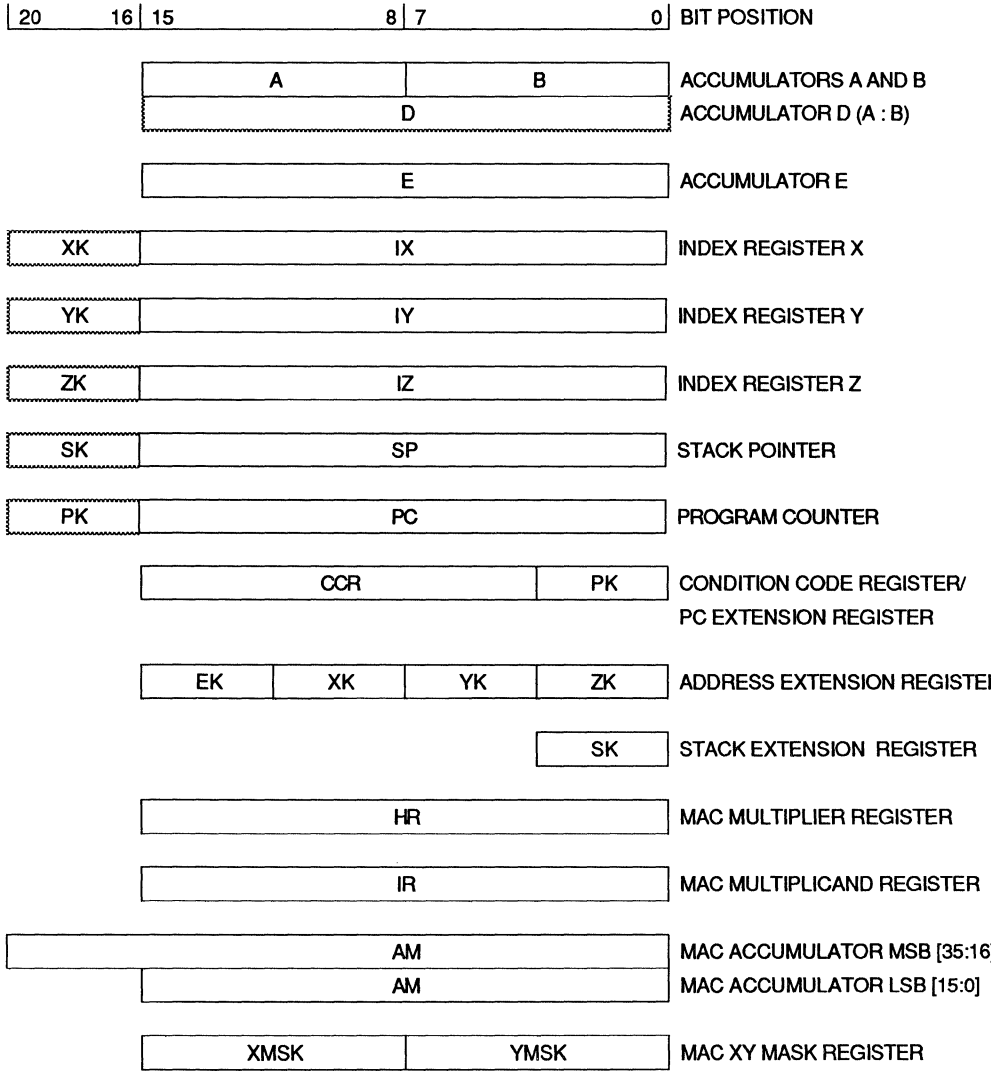


Figure 5–1. CPU16 Register Model

5.1.1 Accumulators

The CPU16 has two 8-bit accumulators (A and B) and one 16-bit accumulator (E). In addition, accumulators A and B can be concatenated into a second 16-bit double accumulator (D).

Accumulators A, B, and D are general-purpose registers used to hold operands and results during mathematic and data manipulation operations.

Accumulator E can be used in the same way as accumulator D, and also extends CPU16 capabilities. It allows more data to be held within the CPU16 during operations, simplifies 32-bit arithmetic and digital signal processing, and provides a practical 16-bit accumulator offset indexed addressing mode.

5.1.2 Index Registers

The CPU16 has three 16-bit index registers (IX, IY, and IZ). Each index register has an associated 4-bit extension field (XK, YK, and ZK).

Concatenated registers and extension fields provide 20-bit indexed addressing and support data structure functions anywhere in the CPU16 address space.

IX and IY can perform the same operations as MC68HC11 registers of the same names, but the CPU16 instruction set provides additional indexed operations.

IZ can perform the same operations as IX and IY, and also provides an additional indexed addressing capability that replaces MC68HC11 direct addressing mode. Initial IZ and ZK extension field values are included in the RESET exception vector, so that ZK : IZ can be used as a direct page pointer out of reset.

5.1.3 Stack Pointer

The CPU16 stack pointer (SP) is 16 bits wide. An associated 4-bit extension field (SK) provides 20-bit stack addressing.

Stack implementation in the CPU16 is from high to low memory. The stack grows downward as it is filled. SK : SP are decremented each time data is pushed on the stack, and incremented each time data is pulled from the stack.

SK : SP point to the next available stack address, rather than to the address of the latest stack entry. Although the stack pointer is normally incremented or decremented by word address, it is possible to push and pull byte-sized data. Setting the stack pointer to an odd value causes misalignment, which affects performance.

5.1.4 Program Counter

The CPU16 program counter (PC) is 16 bits wide. An associated 4-bit extension field (PK) provides 20-bit program addressing.

CPU16 instructions are fetched from even word boundaries. PC0 always has a value of 0, to assure that instruction fetches are made from word-aligned addresses.

5.1.5 Condition Code Register

The 16-bit condition code register can be divided into two functional blocks. The 8 MSB, which correspond to the CCR in the MC68HC11, contain the low-power stop control bit and processor status flags. The 8 LSB contain the interrupt priority field, the DSP saturation mode control bit, and the program counter address extension field.

Figure 5–2 shows the condition code register. Detailed descriptions of each status indicator and field in the register follow the figure.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S	MV	H	EV	N	Z	V	C	IP			SM	PK			

Figure 5–2. Condition Code Register

S — STOP Enable
0 = Stop clock when LPSTOP instruction is executed.
1 = Perform NOP when LPSTOP instruction is executed.

MV — Accumulator M overflow flag
Set when overflow into AM35 has occurred.

H — Half Carry Flag
Set when a carry from A3 or B3 occurs during BCD addition.

EV — Extension Bit Overflow Flag
Set when an overflow into AM31 has occurred.

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 twos complement overflow occurs as the result of an operation.

C — Carry Flag
Set when carry or borrow occurs during arithmetic operation. Also used during shift and rotate to facilitate multiple word operations.

IP[2:0] — Interrupt Priority Field
The priority value in this field (0 to 7) is used to mask interrupts.

SM — Saturate Mode Bit
When SM is set, if either EV or MV is set, data read from AM using TMER or TMET will be given maximum positive or negative value, depending on the state of the AM sign bit before overflow.

PK[3:0] — Program Counter Address Extension Field
This field is concatenated with the program counter to form a 20-bit address.

5.1.6 Address Extension Register and Address Extension Fields

There are six 4-bit address extension fields. EK, XK, YK, and ZK are contained by the address extension register, PK is part of the CCR, and SK stands alone.

Extension fields are the bank portions of 20-bit concatenated bank : byte addresses used in the CPU16 pseudolinear memory management scheme.

All extension fields except EK correspond directly to a register. XK, YK, and ZK extend registers IX, IY, and IZ; PK extends the PC; and SK extends the SP. EK holds the 4 MSB of the 20-bit address used by extended addressing mode.

5.1.7 Multiply and Accumulate Registers

The multiply and accumulate (MAC) registers are part of a CPU submodule that performs repetitive signed fractional multiplication and stores the cumulative result. These operations are part of control-oriented digital signal processing.

There are four MAC registers. Register H contains the 16-bit signed fractional multiplier. Register I contains the 16-bit signed fractional multplicand. Accumulator M is a specialized 36-bit product accumulation register. XMSK and YMSK contain 8-bit mask values used in modulo addressing.

The CPU16 has a special subset of signal processing instructions that manipulate the MAC registers and perform signal processing calculation.

5.2 Memory Management

The CPU16 uses bank switching to provide a 1 Megabyte address space. There are 16 banks within the address space. Each bank is made up of 64 Kbytes addressed from $0000 to $FFFF. Banks are selected by means of address extension fields associated with individual CPU16 registers.

In addition, address space can be split into discrete 1 Megabyte program and data spaces by externally decoding the SIM function code outputs. When this technique is used, instruction fetches and reset vector fetches access program space, while exception vector fetches (other than for reset), data accesses, and stack accesses are made in data space.

5.2.1 Address Extension

All CPU16 resources that are used to generate addresses are effectively 20 bits wide. These resources include extended index registers, program counter, and stack pointer. All addressing modes use 20-bit addresses.

20-bit addresses are formed from a 16-bit byte address generated by an individual CPU16 register and a 4-bit bank address contained in an associated extension field. The byte address corresponds to ADDR[15:0] and the bank address corresponds to ADDR[19:16].

5.2.2 Extension Fields

The six address extension fields are each used in a different type of access. All but EK are associated with particular CPU16 registers. There are a number of ways to manipulate extension fields and the address map. Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for detailed information.

5.3 Data Types

The CPU16 uses the following types of data:

- Bits
- 4-bit signed integers
- 8-bit (byte) signed and unsigned integers
- 8-bit, 2-digit binary coded decimal numbers
- 16-bit (word) signed and unsigned integers
- 32-bit (long word) signed and unsigned integers
- 16-bit signed fractions
- 32-bit signed fractions
- 36-bit signed fixed-point numbers
- 20-bit effective address consisting of 16-bit byte address and 4-bit extension

There are 8 bits in a byte, 16 bits in a word. Bit set and clear instructions use both byte and word operands. Bit test instructions use byte operands.

Negative integers are represented in twos-complement form. Four-bit signed integers, packed two to a byte, are used only as X and Y offsets in MAC and RMAC operations. Thirty-two-bit integers are used only by extended multiply and divide instructions, and by the associated LDED and STED instructions.

Binary coded decimal numbers are packed, two digits per byte. BCD operations use byte operands.

16-bit fractions are used in both fractional multiplication and division, and as multiplicand and multiplier operands in the MAC unit. Bit 15 is the sign bit. There is an implied radix point between bits 15 and 14. There are 15 bits of magnitude — the range of values is –1 ($8000) to $1 - 2^{-15}$ ($7FFF).

Signed 32-bit fractions are used only by fractional multiplication and division instructions. Bit 31 is the sign bit. An implied radix point lies between bits 31 and 30. There are 31 bits of magnitude — the range of values is –1 ($80000000) to $1 - 2^{-31}$ ($7FFFFFFF).

Signed 36-bit fixed-point numbers are used only by the MAC unit. Bit 35 is the sign bit. Bits [34:31] are sign extension bits. There is an implied radix point between bits 31 and 30. There are 31 bits of magnitude, but use of the extension bits allows representation of numbers in the range –16 ($800000000) to 15.999969482 ($7FFFFFFFF).

20-bit addresses are formed by combining a 16-bit byte address with a 4-bit address extension.

5.4 Memory Organization

Both program and data memory are divided into sixteen 64-Kbyte banks. Addressing is pseudolinear — a 20-bit extended address can access any byte location in the appropriate address space.

A word is composed of two consecutive bytes. A word address is normally an even byte address. Byte 0 of a word has a lower 16-bit address than Byte 1. Long words and 32-bit signed fractions consist of two consecutive words, and are normally accessed at the address of Byte 0 in Word 0.

Instruction fetches always access word addresses. Word operands are normally accessed at even byte addresses, but may be accessed at odd byte addresses, with a substantial performance penalty.

To be compatible with the MC68HC11, misaligned word transfers and misaligned stack accesses are allowed. Transferring a misaligned word requires two successive byte transfer operations.

Figure 5–3 shows how each CPU16 data type is organized in memory. Consecutive even addresses show size and alignment.

Address	Type															
$0000	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
$0002	BYTE0								BYTE1							
$0004	±	X OFFSET			±	Y OFFSET			±	X OFFSET			±	Y OFFSET		
$0006	BCD1				BCD0				BCD1				BCD0			
$0008	WORD 0															
$000A	WORD1															
$000C	MSW LONG WORD 0															
$000E	LSW LONG WORD 0															
$0010	MSW LONG WORD 1															
$0012	LSW LONG WORD 1															
$0014	±	⇐ (Radix Point)	16-BIT SIGNED FRACTION 0													
$0016	±	⇐ (Radix Point)	16-BIT SIGNED FRACTION 1													
$0018	±	⇐ (Radix Point)	MSW 32-BIT SIGNED FRACTION 0													
$001A	LSW 32-BIT SIGNED FRACTION 0															0
$001C	±	⇐ (Radix Point)	MSW 32-BIT SIGNED FRACTION 1													
$001E	LSW 32-BIT SIGNED FRACTION 1															0

MAC Data Types

35			32	31		16
±	«	«	«	«	⇐ (Radix Point)	MSW 32-BIT SIGNED FRACTION
				15		0
				LSW 32-BIT SIGNED FRACTION		
				±	⇐ (Radix Point)	16-BIT SIGNED FRACTION

Address Data Type

19	16	15	0
4-Bit Extension		16-Bit Address	

Figure 5–3. Data Types and Memory Organization

5.5 Addressing Modes

The CPU16 uses 10 basic types of addressing. There are one or more addressing modes within each type. Table 5–1 shows the addressing modes.

Table 5–1. Addressing Modes

Mode	Mnemonic	Description
Accumulator Offset	E,X	Index Register X with Accumulator E offset
	E,Y	Index Register Y with Accumulator E offset
	E,Z	Index Register Z with Accumulator E offset
Extended	EXT	Extended
	EXT20	20-bit Extended
Immediate	IMM8	8-bit Immediate
	IMM16	16-bit Immediate
Indexed 8-Bit	IND8, X	Index Register X with unsigned 8-bit offset
	IND8, Y	Index Register Y with unsigned 8-bit offset
	IND8, Z	Index Register Z with unsigned 8-bit offset
Indexed 16-Bit	IND16, X	Index Register X with signed 16-bit offset
	IND16, Y	Index Register Y with signed 16-bit offset
	IND16, Z	Index Register Z with signed 16-bit offset
Indexed 20-Bit	IND20, X	Index Register X with signed 20-bit offset
	IND20, Y	Index Register Y with signed 20-bit offset
	IND20, Z	Index Register Z with signed 20-bit offset
Inherent	INH	Inherent
Post-Modified Index	IXP	Signed 8-bit offset added to Index Register X after effective address is used
Relative	REL8	8-bit relative
	REL16	16-bit relative

All modes generate ADDR[15:0]. This address is combined with ADDR[19:16] from an operand or an extension field to form a 20-bit effective address.

NOTE

Bank switching is transparent to most instructions. ADDR[19:16] of the effective address are changed to make an access across a bank boundary. However, extension field values do not change as a result of effective address computation.

5.5.1 Immediate Addressing Modes

In the immediate modes, an argument is contained in a byte or word immediately following the instruction. For IMM8 and IMM16 modes, the effective address is the address of the argument.

There are three specialized forms of IMM8 addressing.

The AIS, AIX/Y/Z, ADDD and ADDE instructions decrease execution time by sign-extending the 8-bit immediate operand to 16 bits, then adding it to an appropriate register.

The MAC and RMAC instructions use an 8-bit immediate operand to specify two signed 4-bit index register offsets.

The PSHM and PULM instructions use an 8-bit immediate mask operand to indicate which registers must be pushed to or pulled from the stack.

5

5.5.2 Extended Addressing Modes

Regular extended mode instructions contain ADDR[15:0] in the word following the opcode. The effective address is formed by concatenating the EK field and the 16-bit byte address. EXT20 mode is used only by the JMP and JSR instructions. These instructions contain a 20-bit effective address that is zero-extended to 24 bits to give the instruction an even number of bytes.

5.5.3 Indexed Addressing Modes

In the indexed modes, registers IX, IY, and IZ, together with their associated extension fields, are used to calculate the effective address.

For 8-bit indexed modes an 8-bit unsigned offset contained in the instruction is added to the value contained in an index register and its extension field.

For 16-bit modes, a 16-bit signed offset contained in the instruction is added to the value contained in an index register and its extension field.

For 20-bit modes, a 20-bit signed offset (zero-extended to 24 bits) is added to the value contained in an index register. These modes are used for JMP and JSR instructions only.

5.5.4 Inherent Addressing Mode

Inherent mode instructions use information directly available to the processor to determine the effective address. Operands (if any) are system resources and are thus not fetched from memory.

5.5.5 Accumulator Offset Addressing Mode

Accumulator offset modes form an effective address by sign-extending the content of accumulator E to 20 bits, then adding the result to an index register and its associated extension field. This mode allows use of an index register and an accumulator within a loop without corrupting accumulator D.

5.5.6 Relative Addressing Modes

Relative modes are used for branch and long branch instructions. If a branch condition is satisfied, a byte or word signed twos-complement offset is added to the concatenated PK field and program counter. The new PK : PC value is the effective address.

5.5.7 Post-Modified Index Addressing Mode

Post-modified index mode is used by the MOVB and MOVW instructions. A signed 8-bit offset is added to index register X after the effective address formed by XK : IX is used.

5.5.8 Use of HC16 Indexed Mode to Replace HC11 Direct Mode

In MC68HC11 systems, the direct addressing mode can be used to perform rapid accesses to RAM or I/O mapped into bank 0 ($0000 to $00FF), but the CPU16 uses the first 512 bytes of bank 0 for exception vectors. To provide an enhanced replacement for direct mode, the ZK field and index register Z have been assigned reset initialization vectors — by resetting the ZK field to a chosen page, and using indexed mode addressing, a programmer can access useful data structures anywhere in the address map.

5.6 Instruction Set

The instruction set is based upon that of the MC68HC11, but the opcode map has been rearranged to maximize performance with a 16-bit data bus. Much MC68HC11 code can run on the CPU16 following reassembly. The user must take into account changed instruction times, the interrupt mask, and the new interrupt stack frame.

5.6.1 Data Movement Instructions

The CPU16 has a complete set of 8 and 16-bit data movement instructions, as well as instructions to support 32-bit intermodule bus (IMB) operations. General-purpose load, store, transfer and move instructions facilitate movement of data to and from memory and peripherals. Special purpose instructions enhance indexing, extended addressing, stacking, and digital signal processing.

5.6.1.1 Load Instructions

Load instructions copy memory content into an accumulator or register. Memory content is not changed by the operation.

There are specialized load instructions for stacking, indexing, extended addressing, and digital signal processing. Refer to the appropriate summary for more information.

Table 5–2. Load Summary

Mnemonic	Function	Operation
LDAA	Load A	(M) ⇒ A
LDAB	Load B	(M) ⇒ B
LDD	Load D	(M : M + 1) ⇒ D
LDE	Load E	(M : M + 1) ⇒ E
LDED	Load Concatenated E and D	(M : M + 1) ⇒ E (M + 2 : M + 3) ⇒ D

5.6.1.2 Move Instructions

These instructions move data bytes or words from one location to another in memory.

Table 5–3. Move Summary

Mnemonic	Function	Operation
MOVB	Move Byte	$(M_1) \Rightarrow M_2$
MOVW	Move Word	$(M : M + 1_1) \Rightarrow M : M + 1_2$

5.6.1.3 Store Instructions

Store instructions copy the content of an accumulator or register to memory. Register/accumulator content is not changed by the operation.

There are specialized store instructions for indexing, extended addressing, and CCR manipulation. Refer to the appropriate summary for more information.

Table 5–4. Store Summary

Mnemonic	Function	Operation
STAA	Store A	(A) ⇒ M
STAB	Store B	(B) ⇒ M
STD	Store D	(D) ⇒ M : M + 1
STE	Store E	(E) ⇒ M : M + 1
STED	Store Concatenated D and E	(E) ⇒ M : M + 1 (D) ⇒ M + 2 : M + 3

5.6.1.4 Transfer Instructions

These instructions transfer the content of a register or accumulator to another register or accumulator. Content of the source is not changed by the operation.

There are specialized transfer instructions for stacking, indexing, extended addressing, CCR manipulation, and digital signal processing. Refer to the appropriate summary for more information.

Table 5–5. Transfer Summary

Mnemonic	Function	Operation
TAB	Transfer A to B	(A) ⇒ B
TBA	Transfer B to A	(B) ⇒ A
TDE	Transfer D to E	(D) ⇒ E
TED	Transfer E to D	(E) ⇒ D

5.6.1.5 Exchange Instructions

These instructions exchange the contents of pairs of registers or accumulators. There are specialized exchange instructions for indexing.

Table 5–6. Exchange Summary

Mnemonic	Function	Operation
XGAB	Exchange A with B	(A) ⇔ (B)
XGDE	Exchange D with E	(D) ⇔ (E)

5.6.2 Mathematic Instructions

5

The CPU16 has a full set of 8- and 16-bit mathematic instructions. There are instructions for signed and unsigned arithmetic, division and multiplication, as well as a complete set of Boolean operators.

Special arithmetic and logic instructions aid stacking operations, indexing, extended addressing, BCD calculation, and condition code register manipulation. There are also dedicated multiply and accumulate instructions.

5.6.2.1 Addition and Subtraction Instructions

Signed and unsigned 8- and 16-bit arithmetic instructions can be performed between registers or between registers and memory. Instructions that add or subtract the value of the CCR carry bit facilitate multiple precision computation.

Table 5–7. Addition Summary

Mnemonic	Function	Operation
ABA	Add B to A	(A) + (B) ⇒ A
ADCA	Add with Carry to A	(A) + (M) + C ⇒ A
ADCB	Add with Carry to B	(B) + (M) + C ⇒ B
ADCD	Add with Carry to D	(D) + (M : M + 1) + C ⇒ D
ADCE	Add with Carry to E	(E) + (M : M + 1) + C ⇒ E
ADDA	Add to A	(A) + (M) ⇒ A
ADDB	Add to B	(B) + (M) ⇒ B
ADDD	Add to D	(D) + (M : M + 1) ⇒ D
ADDE	Add to E	(E) + (M : M + 1) ⇒ E
ADE	Add D to E	(E) + (D) ⇒ E

Table 5–8. Subtraction Summary

SBA	Subtract B from A	(A) – (B) ⇒ A
SBCA	Subtract with Carry from A	(A) – (M) – C ⇒ A
SBCB	Subtract with Carry from B	(B) – (M) – C ⇒ B
SBCD	Subtract with Carry from D	(D) – (M : M + 1) – C ⇒ D
SBCE	Subtract with Carry from E	(E) – (M : M + 1) – C ⇒ E
SDE	Subtract D from E	(E) – (D) ⇒ E
SUBA	Subtract from A	(A) – (M) ⇒ A
SUBB	Subtract from B	(B) – (M) ⇒ B
SUBD	Subtract from D	(D) – (M : M + 1) ⇒ D
SUBE	Subtract from E	(E) – (M : M + 1) ⇒ E

5

5.6.2.2 Binary Coded Decimal Instructions

To add binary coded decimal operands, use addition instructions that set the half-carry bit in the CCR, then adjust the result with the DAA instruction.

Table 5–9. BCD Summary

ABA	Add B to A	(A) + (B) ⇒ A
ADCA	Add with Carry to A	(A) + (M) + C ⇒ A
ADCB	Add with Carry to B	(B) + (M) + C ⇒ B
ADDA	Add to A	(A) + (M) ⇒ A
ADDB	Add to B	(B) + (M) ⇒ B
DAA	Decimal Adjust A	$(A)_{10}$
SXT	Sign Extend B into A	If B7 = 1 then A = $FF else A = $00

5.6.2.3 Compare and Test Instructions

Compare and test instructions perform subtraction between a pair of registers or between a register and memory. The result is not stored, but condition codes are set by the operation. These instructions are generally used to establish conditions for branch instructions.

Table 5–10. Compare and Test Summary

CBA	Compare A to B	(A) – (B)
CMPA	Compare A to Memory	(A) – (M)
CMPB	Compare B to Memory	(B) – (M)
CPD	Compare D to Memory	(D) – (M : M + 1)
CPE	Compare E to Memory	(E) – (M : M + 1)
TST	Test for Zero or Minus	(M) – $00
TSTA	Test A for Zero or Minus	(A) – $00
TSTB	Test B for Zero or Minus	(B) – $00
TSTD	Test D for Zero or Minus	(D) – $0000
TSTE	Test E for Zero or Minus	(E) – $0000
TSTW	Test for Zero or Minus Word	(M : M + 1) – $0000

5.6.2.4 Multiplication and Division Instructions

There are instructions for signed and unsigned 8- and 16-bit multiplication, as well as for signed 16-bit fractional multiplication. Eight-bit multiplication operations have a 16-bit product. Sixteen-bit multiplication operations can have either 16- or 32-bit products.

All division operations have 16-bit divisors, but dividends can be either 16- or 32-bit numbers. Quotients and remainders of all division operations are 16-bit numbers. There are instructions for signed and unsigned division, as well as for fractional division.

Fractional multiplication uses 16-bit operands. Bit 15 is the sign bit. There is an implied radix point between bits 15 and 14. The range of values is –1 ($8000) to 0.999969482 ($7FFF). The MSB of the result is its sign bit, and there is an implied radix point between the sign bit and the rest of the result.

There are special 36-bit signed fractional multiply and accumulate unit instructions to support digital signal processing operations. Refer to the appropriate summary for more information.

Table 5–11. Multiplication and Division Summary

EDIV	Extended Unsigned Divide	(E : D) / (IX) Quotient ⇒ IX Remainder ⇒ D
EDIVS	Extended Signed Divide	(E : D) / (IX) Quotient ⇒ IX Remainder ⇒ D
EMUL	Extended Unsigned Multiply	(E) * (D) ⇒ E : D
EMULS	Extended Signed Multiply	(E) * (D) ⇒ E : D
FDIV	Fractional Divide	(D) / (IX) ⇒ IX Remainder ⇒ D
FMULS	Fractional Signed Multiply	(E) * (D) ⇒ E : D
IDIV	Integer Divide	(D) / (IX) ⇒ IX; Remainder ⇒ D
MUL	Multiply	(A) * (B) ⇒ D

5

5.6.2.5 Decrement and Increment Instructions

These instructions are optimized 8- and 16-bit addition and subtraction operations. Because they do not affect the carry bit in the CCR, they are particularly well suited for loop counters in multiple-precision computation routines.

Table 5–12. Decrement and Increment Summary

DEC	Decrement Memory	(M) – $01 ⇒ M
DECA	Decrement A	(A) – $01 ⇒ A
DECB	Decrement B	(B) – $01 ⇒ B
DECW	Decrement Memory Word	(M : M + 1) – $0001 ⇒ M : M + 1
INC	Increment Memory	(M) + $01 ⇒ M
INCA	Increment A	(A) + $01 ⇒ A
INCB	Increment B	(B) + $01 ⇒ B
INCW	Increment Memory Word	(M : M + 1) + $0001 ⇒ M : M + 1

5.6.2.6 Clear, Complement and Negate Instructions

Each of these instructions performs a specific binary operation on a value in an accumulator or in memory. Clear operations set the value to 0, complement operations replace the value with its ones complement, and negate operations replace the value with its twos complement.

Table 5–13. Clear, Complement and Negate Summary

CLR	Clear Memory	$00 ⇒ M
CLRA	Clear A	$00 ⇒ A
CLRB	Clear B	$00 ⇒ B
CLRD	Clear D	$0000 ⇒ D
CLRE	Clear E	$0000 ⇒ E
CLRW	Clear Memory Word	$0000 ⇒ M : M + 1
COM	Ones Complement Byte	$FF – (M) ⇒ M
COMA	Ones Complement A	$FF – (A) ⇒ A
COMB	Ones Complement B	$FF – (B) ⇒ B
COMD	Ones Complement D	$FFFF – (D) ⇒ D
COME	Ones Complement E	$FFFF – (E) ⇒ E
COMW	Ones Complement Word	$FFFF - M : M + 1 ⇒ M : M + 1
NEG	Twos Complement Byte	$00 – (M) ⇒ M
NEGA	Twos Complement A	$00 – (A) ⇒ A
NEGB	Twos Complement B	$00 – (B) ⇒ B
NEGD	Twos Complement D	$0000 – (D) ⇒ D
NEGE	Twos Complement E	$0000 – (E) ⇒ E
NEGW	Twos Complement Word	$0000 – (M : M + 1) ⇒ M : M + 1

5.6.2.7 Boolean Logic Instructions

Each of these instructions performs the Boolean logic operation represented by the mnemonic. There are 8- and 16-bit versions of each instruction.

There are special forms of logic instructions for stack pointer, program counter, index register, and address extension field manipulation. Refer to the appropriate summary for more information.

Table 5–14. Boolean Logic Summary

Mnemonic	Function	Operation
ANDA	AND A	(A) • (M) ⇒ A
ANDB	AND B	(B) • (M) ⇒ B
ANDD	AND D	(D) • (M : M + 1) ⇒ D
ANDE	AND E	(E) • (M : M + 1) ⇒ E
EORA	Exclusive OR A	(A) ⊕ (M) ⇒ A
EORB	Exclusive OR B	(B) ⊕ (M) ⇒ B
EORD	Exclusive OR D	(D) ⊕ (M : M + 1) ⇒ D
EORE	Exclusive OR E	(E) ⊕ (M : M + 1) ⇒ E
ORAA	OR A	(A) + (M) ⇒ A
ORAB	OR B	(B) + (M) ⇒ B
ORD	OR D	(D) + (M : M + 1) ⇒ D
ORE	OR E	(E) + (M : M + 1) ⇒ E

5

5.6.3 Bit Test and Manipulation Instructions

These operations use a mask value to test or change the value of individual bits in an accumulator or in memory. BITA and BITB provide a convenient means of setting condition codes without altering the value of either operand.

Table 5–15. Bit Test and Manipulation Summary

Mnemonic	Function	Operation
BITA	Bit Test A	(A) • (M)
BITB	Bit Test B	(B) • (M)
BCLR	Clear Bit(s)	(M) • $\overline{(\text{Mask})}$ ⇒ M
BCLRW	Clear Bit(s) Word	(M : M + 1) • $\overline{(\text{Mask})}$ ⇒ M : M + 1
BSET	Set Bit(s)	(M) + (Mask) ⇒ M
BSETW	Set Bit(s) Word	(M : M + 1) + (Mask) ⇒ M : M + 1

5.6.4 Shift and Rotate Instructions

There are shift and rotate commands for all accumulators, for memory bytes and for memory words. All shift and rotate operations pass the shifted-out bit through the carry bit in the CCR in order to facilitate multiple-byte and multiple-word operations. There are no separate logical left shift operations — use ASL instead. Motorola assemblers assemble LSL mnemonics as ASL operations.

Table 5–16. Logic Shift Summary

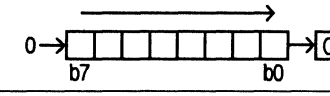
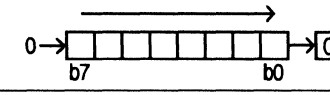
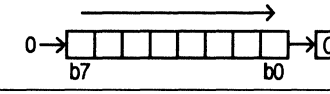
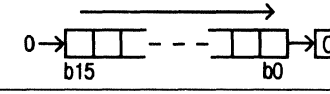
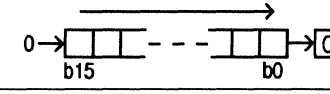
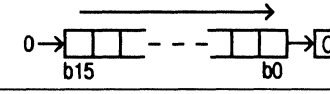
LSR	Logic Shift Right	
LSRA	Logic Shift Right A	
LSRB	Logic Shift Right B	
LSRD	Logic Shift Right D	
LSRE	Logic Shift Right E	
LSRW	Logic Shift Right Word	

Table 5–17. Arithmetic Shift Summary

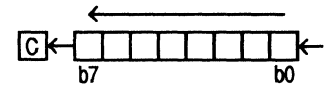
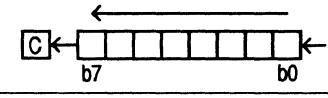
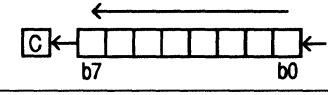
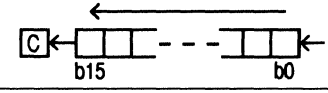
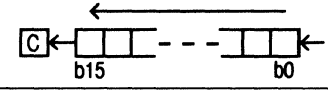
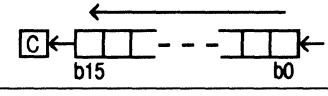
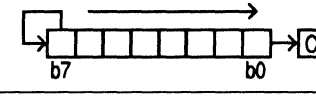
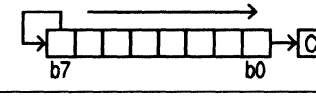
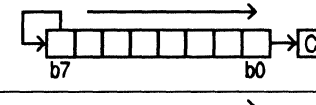
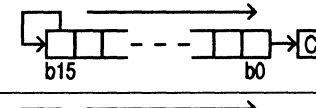
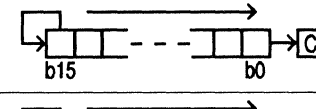
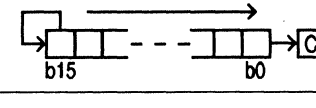
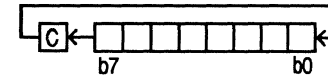
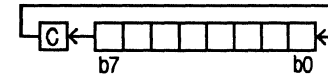
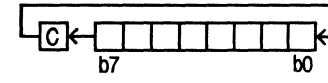
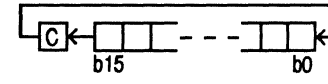
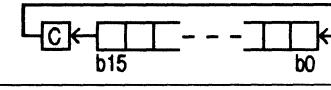
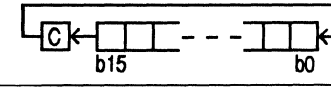
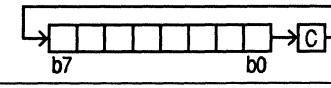
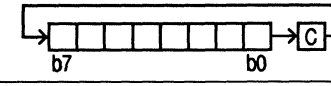
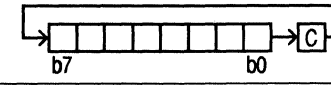
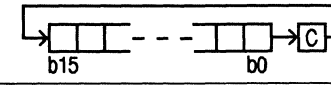
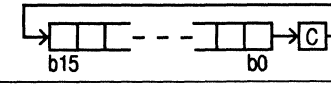
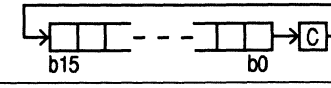
Mnemonic	Function	Operation
ASL (LSL)	Arithmetic Shift Left	
ASLA (LSLA)	Arithmetic Shift Left A	
ASLB (LSLB)	Arithmetic Shift Left B	
ASLD (LSLD)	Arithmetic Shift Left D	
ASLE (LSLE)	Arithmetic Shift Left E	
ASLW (LSLW)	Arithmetic Shift Left Word	
ASR	Arithmetic Shift Right	
ASRA	Arithmetic Shift Right A	
ASRB	Arithmetic Shift Right B	
ASRD	Arithmetic Shift Right D	
ASRE	Arithmetic Shift Right E	
ASRW	Arithmetic Shift Right Word	

Table 5–18. Rotate Summary

ROL	Rotate Left	
ROLA	Rotate Left A	
ROLB	Rotate Left B	
ROLD	Rotate Left D	
ROLE	Rotate Left E	
ROLW	Rotate Left Word	
ROR	Rotate Right	
RORA	Rotate Right A	
RORB	Rotate Right B	
RORD	Rotate Right D	
RORE	Rotate Right E	
RORW	Rotate Right Word	

5.6.5 Program Control Instructions

Program control instructions affect the sequence of instruction execution.

Branch instructions cause sequence to change when specific conditions exist. The CPU16 has short, long, and bit-condition branches.

Jump instructions cause immediate changes in sequence. The CPU16 has a true 20-bit address jump instruction.

Subroutine instructions optimize the process of temporarily transferring control to a segment of code that performs a particular task. The CPU16 can branch or jump to subroutines.

Interrupt instructions handle immediate transfer of control to a routine that performs a critical task. Software interrupts are a type of exception.

5.6.5.1 Short Branch Instructions

Short branch instructions operate as follows. When a specified condition is met, a signed 8-bit offset is added to the value in the program counter. If addition causes the value in the PC to be greater than $FFFF or less than $0000, the PK extension field is incremented or decremented. Program execution continues at the new extended address.

Short branch instructions can be classified by the type of condition that must be satisfied in order for a branch to be taken. Some instructions belong to more than one classification.

Unary branch instructions always execute.

Simple branches are taken when a specific bit in the condition code register is in a specific state as a result of a previous operation.

Unsigned conditional branches are taken when comparison or test of unsigned quantities results in a specific combination of condition code register bits.

Signed branches are taken when comparison or test of signed quantities results in a specific combination of condition code register bits.

Table 5–19. Short Branch Summary

Unary Branches			
Mnemonic	**Opcode**	**Equation**	**Condition**
BRA	B0	1 = 1	True
BRN	B1	1 = 0	False
Simple Branches			
Mnemonic	**Opcode**	**Equation**	**Condition**
BCC	B4	C = 0	Equation
BCS	B5	C = 1	Equation
BEQ	B7	Z = 1	Equation
BMI	BB	N = 1	Equation
BNE	B6	Z = 0	Equation
BPL	BA	N = 0	Equation
BVC	B8	V = 0	Equation
BVS	B9	V = 1	Equation
Unsigned Branches			
Mnemonic	**Opcode**	**Equation**	**Condition**
BCC	B4	C = 0	(X) ≥ (M)
BCS	B5	C = 1	(X) < (M)
BEQ	B7	Z = 1	(X) = (M)
BHI	B2	C + Z = 0	(X) > (M)
BLS	B3	C + Z = 1	(X) ≤ (M)
BNE	B6	Z = 0	(X) ≠ (M)
Signed Branches			
Mnemonic	**Opcode**	**Equation**	**Condition**
BEQ	B7	Z = 1	(X) = (M)
BGE	BC	N ⊕ V = 0	(X) ≥ (M)
BGT	BE	Z + (N ⊕ V) = 0	(X) > (M)
BLE	BF	Z + (N ⊕ V) = 1	(X) ≤ (M)
BLT	BD	N ⊕ V = 1	(X) < (M)
BNE	B6	Z = 0	(X) ≠ (M)

The numeric range of short branch offset values is $80 (–128) to $7F (127), but actual displacement from the instruction differs from the range for two reasons. First, PC values are automatically aligned to word boundaries. Only even offsets are valid — an odd offset value is rounded down. Maximum positive offset is $7E. Second, instruction pipelining affects the value in the PC at the time an instruction executes. The value to which the offset is added is the address of the instruction plus $0006. At maximum positive offset ($7E), displacement from the branch instruction is 132. At maximum negative offset ($80), displacement is –122.

5.6.5.2 Long Branch Instructions

Long branch instructions operate as follows. When a specified condition is met, a signed 16-bit offset is added to the value in the program counter. If addition causes the value in the PC to be greater than $FFFF or less than $0000, the PK extension field is incremented or decremented. Program execution continues at the new extended address.

Long branch instructions can be classified by the type of condition that must be satisfied in order for a branch to be taken. Some instructions belong to more than one classification.

Unary branch instructions always execute.

Simple branches are taken when a specific bit in the condition code register is in a specific state as a result of a previous operation.

Unsigned branches are taken when comparison or test of unsigned quantities results in a specific combination of condition code register bits.

Signed branches are taken when comparison or test of signed quantities results in a specific combination of condition code register bits.

Table 5–20. Long Branch Instructions

Unary Branches			
Mnemonic	**Opcode**	**Equation**	**Condition**
LBRA	3780	1 = 1	True
LBRN	3781	1 = 0	False
Simple Branches			
Mnemonic	**Opcode**	**Equation**	**Condition**
LBCC	3784	C = 0	Equation
LBCS	3785	C = 1	Equation
LBEQ	3787	Z = 1	Equation
LBEV	3791	EV = 1	Equation
LBMI	378B	N = 1	Equation
LBMV	3790	MV = 1	Equation
LBNE	3786	Z = 0	Equation
LBPL	378A	N = 0	Equation
LBVC	3788	V = 0	Equation
LBVS	3789	V = 1	Equation
Unsigned Branches			
Mnemonic	**Opcode**	**Equation**	**Condition**
LBCC	3784	C = 0	(X) ≥ (M)
LBCS	3785	C = 1	(X) < (M)
LBEQ	3787	Z = 1	(X) = (M)
LBHI	3782	C + Z = 0	(X) > (M)
LBLS	3783	C + Z = 1	(X) ≤ (M)
LBNE	3786	Z = 0	(X) ≠ (M)
Signed Branches			
Mnemonic	**Opcode**	**Equation**	**Condition**
LBEQ	3787	Z = 1	(X) = (M)
LBGE	378C	N ⊕ V = 0	(X) ≥ (M)
LBGT	378E	Z + (N ⊕ V) = 0	(X) > (M)
LBLE	378F	Z + (N ⊕ V) = 1	(X) ≤ (M)
LBLT	378D	N ⊕ V = 1	(X) < (M)
LBNE	3786	Z = 0	(X) ≠ (M)

The numeric range of long branch offset values is $8000 (–32768) to $7FFF (32767), but actual displacement from the instruction differs from the range for two reasons. First, PC values are automatically aligned to word boundaries. Only even offsets are valid — an odd offset value will be rounded down. Maximum positive offset is $7FFE. Second, instruction pipelining affects the value in the PC at the time an instruction executes. The value to which the offset is added is the address of the instruction plus $0006. At maximum positive offset ($7FFE), displacement from the instruction is 32772. At maximum negative offset ($8000), displacement is –32762.

5.6.5.3 Bit Condition Branch Instructions

Bit condition branches are taken when specific bits in a memory byte are in a specific state. A mask operand is used to test a memory location pointed to by a 20-bit indexed or extended effective address. If the bits in memory match the mask, an 8- or 16-bit signed relative offset is added to the current value of the program counter. If addition causes the value in the PC to be greater than $FFFF or less than $0000, the PK extension field is incremented or decremented. Program execution continues at the new extended address.

Table 5–21. Bit Condition Branch Summary

Mnemonic	Addressing Mode	Opcode	Equation
BRCLR	IND8, X	CB	(M) • (Mask) = 0
	IND8, Y	DB	
	IND8, Z	EB	
	IND16, X	0A	
	IND16, Y	1A	
	IND16, Z	2A	
	EXT	3A	
BRSET	IND8, X	8B	$(\overline{M})$ • (Mask) = 0
	IND8, Y	9B	
	IND8, Z	AB	
	IND16, X	0B	
	IND16, Y	1B	
	IND16, Z	2B	
	EXT	3B	

The numeric range of 8-bit offset values is $80 (–128) to $7F (127), and the numeric range of 16-bit offset values is $8000 (–32768) to $7FFF (32767), but actual displacement from the branch instruction differs from the range for two reasons. First, PC values are automatically aligned to word boundaries. Only even offsets are valid — an odd offset value is rounded down. Maximum positive 8-bit offset is $7E; maximum positive 16-bit offset is $7FFE. Second, instruction pipelining affects the value in the PC at the time an instruction executes. The value to which the offset is added is the address of the instruction plus $0006. Maximum positive ($7E) and negative ($80) 8-bit offsets correspond to displacements of 132 and –122 from the branch instruction. Maximum positive ($7FFE) and negative ($8000) 16-bit offsets correspond to displacements of 32772 and –32762.

5.6.5.4 Jump Instruction

The CPU16 JMP instruction uses 20-bit addressing, so that control can be passed to any address in the memory map. It should be noted that BRA and LBRA execute in fewer cycles than the indexed forms of JMP.

Table 5–22. Jump Summary

Mnemonic	Function	Operation
JMP	Jump	20-bit Address ⇒ PK : PC

5.6.5.5 Subroutine Instructions

5

Subroutines can be called by short (BSR) or long (LBSR) branches, or by a jump (JSR). A single instruction, RTS, returns control to the calling routine.

All three types of calling instructions stack return PC and CCR values prior to transferring control to a subroutine. Stacking the CCR also saves the PK extension field. Other resources can be saved by means of the PSHM instruction, if necessary.

Table 5–23. Subroutine Summary

Mnemonic	Function	Operation
BSR	Branch to Subroutine	(PK : PC) – 2 ⇒ PK : PC Push (PC) (SK : SP) – 2 ⇒ SK : SP Push (CCR) (SK : SP) – 2 ⇒ SK : SP (PK : PC) + Offset ⇒ PK : PC
JSR	Jump to Subroutine	Push (PC) (SK : SP) – 2 ⇒ SK : SP Push (CCR) (SK : SP) – 2 ⇒ SK : SP 20-bit Address ⇒ PK : PC
LBSR	Long Branch to Subroutine	Push (PC) (SK : SP) – 2 ⇒ SK : SP Push (CCR) (SK : SP) – 2 ⇒ SK : SP (PK : PC) + Offset ⇒ PK : PC
RTS	Return from Subroutine	(SK : SP) + 2 ⇒ SK : SP Pull PK (SK : SP) + 2 ⇒ SK : SP Pull PC (PK : PC) – 2 ⇒ PK : PC

Instruction pipelining affects the operation of BSR. When a subroutine is called, PK : PC contain the address of the calling instruction plus $0006. LBSR and JSR stack this value, but BSR must adjust it prior to stacking. LBSR and JSR are 4-byte instructions. For program execution to resume at the instruction immediately following them, RTS must subtract $0002 from the stacked PK : PC value. BSR is a 2-byte instruction. BSR subtracts $0002 from the stacked value prior to stacking so that RTS will work correctly.

5.6.5.6 Interrupt Instructions

The SWI instruction initiates synchronous exception processing. First, return PC and CCR values are stacked (stacking the CCR saves the PK extension field). After return values are stacked, the PK field is cleared, and the PC is loaded with exception vector 6 (content of address $000C).

The RTI instruction is used to terminate all exception handlers, including interrupt service routines. It causes normal execution to resume with the instruction following the last instruction that executed prior to interrupt.

Table 5–24. Interrupt Summary

Mnemonic	Function	Operation
RTI	Return from Interrupt	(SK : SP) + 2 ⇒ SK : SP Pull CCR (SK : SP) + 2 ⇒ SK : SP Pull PC (PK : PC) – 6 ⇒ PK : PC
SWI	Software Interrupt	(PK : PC) + 2 ⇒ PK : PC Push (PC) (SK : SP) – 2 ⇒ SK : SP Push (CCR) (SK : SP) – 2 ⇒ SK : SP $0 ⇒ PK SWI Vector ⇒ PC

Instruction pipelining affects the operation of SWI. When an interrupt occurs, PK : PC contain the address of the interrupted instruction plus $0006. This value is stacked during asynchronous exception processing, but synchronous exceptions, such as SWI, must adjust the stacked value so that RTI can work correctly. For program execution to resume with the interrupted instruction following an asynchronous interrupt, RTI must subtract $0006 from the stacked PK : PC value. Synchronous interrupts allow an interrupted instruction to finish execution before exception processing begins. The SWI instruction must add $0002 prior to stacking in order for execution to resume correctly.

5.6.6 Indexing and Address Extension Instructions

The CPU16 has a complete set of instructions that enable a user to take full advantage of 20-bit pseudolinear addressing. These instructions use specialized forms of mathematic and data transfer instructions to perform index register manipulation and extension field manipulation.

5.6.6.1 Indexing Instructions

Indexing instructions perform 8- and 16-bit operations on the three index registers and accumulators, other registers, or memory. Index addition and transfer instructions also affect the associated extension field.

Table 5–25. Indexing Summary

Addition Instructions		
Mnemonic	**Function**	**Operation**
ABX	Add B to IX	(XK : IX) + (000 : B) ⇒ XK : IX
ABY	Add B to IY	(YK : IY) + (000 : B) ⇒ YK : IY
ABZ	Add B to IZ	(ZK : Z) + (000 : B) ⇒ ZK : IZ
ADX	Add D to IX	(XK : IX) + (« D) ⇒ XK : IX
ADY	Add D to IY	(YK : IY) + (« D) ⇒ YK : IY
ADZ	Add D to IZ	(ZK : IZ) + (« D) ⇒ ZK : IZ
AEX	Add E to IX	(XK : IX) + (« D)⇒ XK : IX
AEY	Add E to IY	(YK : IY) + (« E) ⇒ YK : IY
AEZ	Add E to IZ	(ZK : IZ) + (« E) ⇒ ZK : IZ
AIX	Add Immediate Value to IX	XK : IX + (« IMM8/16) ⇒ XK : IX
AIY	Add Immediate Value to IY	YK : IY + (« IMM8/16) ⇒ YK : IY
AIZ	Add Immediate Value to IZ	ZK : IZ + (« IMM8/16) ⇒ ZK : IZ
Compare Instructions		
Mnemonic	**Function**	**Operation**
CPX	Compare IX to Memory	(IX) – (M : M + 1)
CPY	Compare IY to Memory	(IY) – (M : M + 1)
CPZ	Compare IZ to Memory	(IZ) – (M : M + 1)
Load Instructions		
Mnemonic	**Function**	**Operation**
LDX	Load IX	(M : M + 1) ⇒ IX
LDY	Load IY	(M : M + 1) ⇒ IY
LDZ	Load IZ	(M : M + 1) ⇒ IZ

Table 5–25. Indexing Summary (Continued)

Store Instructions		
Mnemonic	**Function**	**Operation**
STX	Store IX	(IX) ⇒ M : M + 1
STY	Store IY	(IY) ⇒ M : M + 1
STZ	Store IZ	(IZ) ⇒ M : M + 1
Transfer Instructions		
Mnemonic	**Function**	**Operation**
TSX	Transfer SP to IX	(SK : SP) + 2 ⇒ XK : IX
TSY	Transfer SP to IY	(SK : SP) + 2 ⇒ YK : IY
TSZ	Transfer SP to IZ	(SK : SP) + 2 ⇒ ZK : IZ
TXS	Transfer IX to SP	(XK : IX) – 2 ⇒ SK : SP
TXY	Transfer IX to IY	(XK : IX) ⇒ YK : IY
TXZ	Transfer IX to IZ	(XK : IX) ⇒ ZK : IZ
TYS	Transfer IY to SP	(YK : IY) – 2 ⇒ SK : SP
TYX	Transfer IY to IX	(YK : IY) ⇒ XK : IX
TYZ	Transfer IY to IZ	(YK : IY) ⇒ ZK : IZ
TZS	Transfer IZ to SP	(ZK : IZ) – 2 ⇒ SK : SP
TZX	Transfer IZ to IX	(ZK : IZ) ⇒ XK : IX
TZY	Transfer IZ to IY	(ZK : IZ) ⇒ ZK : IY
Exchange Instructions		
Mnemonic	**Function**	**Operation**
XGDX	Exchange D with IX	(D) ⇔ (IX)
XGDY	Exchange D with IY	(D) ⇔ (IY)
XGDZ	Exchange D with IZ	(D) ⇔ (IZ)
XGEX	Exchange E with IX	(E) ⇔ (IX)
XGEY	Exchange E with IY	(E) ⇔ (IY)
XGEZ	Exchange E with IZ	(E) ⇔ (IZ)

5.6.6.2 Address Extension Instructions

Address extension instructions transfer extension field contents to or from accumulator B. Other types of operations can be performed on the extension field value while it is in the accumulator.

Table 5–26. Address Extension Summary

Mnemonic	Function	Operation
TBEK	Transfer B to EK	(B) ⇒ EK
TBSK	Transfer B to SK	(B) ⇒ SK
TBXK	Transfer B to XK	(B) ⇒ XK
TBYK	Transfer B to YK	(B) ⇒ YK
TBZK	Transfer B to ZK	(B) ⇒ ZK
TEKB	Transfer EK to B	$0 ⇒ B[7:4] (EK) ⇒ B[3:0]
TSKB	Transfer SK to B	(SK) ⇒ B[3:0] $0 ⇒ B[7:4]
TXKB	Transfer XK to B	$0 ⇒ B[7:4] (XK) ⇒ B[3:0]
TYKB	Transfer YK to B	$0 ⇒ B[7:4] (YK) ⇒ B[3:0]
TZKB	Transfer ZK to B	$0 ⇒ B[7:4] (ZK) ⇒ B[3:0]

5.6.7 Stacking Instructions

There are two types of stacking instructions. Stack pointer instructions use specialized forms of mathematic and data transfer instructions to perform stack pointer manipulation. Stack operation instructions save information on and retrieve information from the system stack.

Table 5–27. Stacking Summary

Stack Pointer Instructions		
Mnemonic	**Function**	**Operation**
AIS	Add Immediate Data to SP	SK : SP + (« IMM16) ⇒ SK : SP
CPS	Compare SP to Memory	(SP) – (M : M + 1)
LDS	Load SP	(M : M + 1) ⇒ SP
STS	Store SP	(SP) ⇒ M : M + 1
TSX	Transfer SP to IX	(SK : SP) + 2 ⇒ XK : IX
TSY	Transfer SP to IY	(SK : SP) + 2 ⇒ YK : IY
TSZ	Transfer SP to IZ	(SK : SP) + 2 ⇒ ZK : IZ
TXS	Transfer IX to SP	(XK : IX) – 2 ⇒ SK : SP
TYS	Transfer IY to SP	(YK : IY) – 2 ⇒ SK : SP
TZS	Transfer IZ to SP	(ZK : IZ) – 2 ⇒ SK : SP
Stack Operation Instructions		
Mnemonic	**Function**	**Operation**
PSHA	Push A	(SK : SP) + 1 ⇒ SK : SP Push (A) (SK : SP) – 2 ⇒ SK : SP
PSHB	Push B	(SK : SP) + 1 ⇒ SK : SP Push (B) (SK : SP) – 2 ⇒ SK : SP
PSHM	Push Multiple Registers Mask bits: 0 = D 1 = E 2 = IX 3 = IY 4 = IZ 5 = K 6 = CCR 7 = (reserved)	For mask bits 0 to 6 : If mask bit set Push register (SK:SP) – 2 ⇒ SK : SP
PULA	Pull A	(SK : SP) + 2 ⇒ SK : SP Pull (A) (SK : SP) – 1 ⇒ SK : SP
PULB	Pull B	(SK : SP) + 2 ⇒ SK : SP Pull (B) (SK : SP) – 1 ⇒ SK : SP
PULM	Pull Multiple Registers Mask bits: 0 = CCR[15:4] 1 = K 2 = IZ 3 = IY 4 = IX 5 = E 6 = D 7 = (reserved)	For mask bits 0 to 7: If mask bit set (SK:SP) + 2 ⇒ SK : SP Pull register

5.6.8 Condition Code Instructions

Condition code instructions use specialized forms of mathematic and data transfer instructions to perform condition code register manipulation. Interrupts are not acknowledged until the instruction following ANDP, ORP, TAP, and TDP has executed. Refer to **5.6.10 Stop and Wait Instructions** for more information.

Table 5–28. Condition Code Summary

Mnemonic	Function	Operation
ANDP	AND CCR	(CCR) • IMM16 ⇒ CCR[15:4]
ORP	OR CCR	(CCR) + IMM16 ⇒ CCR[15:4]
TAP	Transfer A to CCR	(A[7:0]) ⇒ CCR[15:8]
TDP	Transfer D to CCR	(D) ⇒ CCR[15:4]
TPA	Transfer CCR MSB to A	(CCR[15:8]) ⇒ A
TPD	Transfer CCR to D	(CCR) ⇒ D

5.6.9 Digital Signal Processing Instructions

DSP instructions use the CPU16 multiply and accumulate unit to implement digital filters and other signal processing functions. Other instructions, notably those that operate on concatenated E and D accumulators, are also used for signal processing.

Table 5–29. DSP Summary

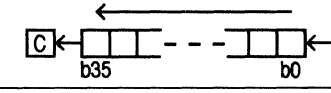
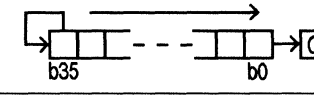
Mnemonic	Function	Operation
ACE	Add E to AM[31:15]	(AM[31:15]) + (E) ⇒ AM
ACED	Add concatenated E and D to AM	(E : D) + (AM) ⇒ AM
ASLM	Arithmetic Shift Left AM	
ASRM	Arithmetic Shift Right AM	
CLRM	Clear AM	$000000000 ⇒ AM[35:0]
LDHI	Initialize HR and IR	(M : M + 1)$_X$ ⇒ HR (M : M + 1)$_Y$ ⇒ IR

Table 5–29. DSP Summary (Continued)

MAC	Multiply and Accumulate Signed 16-Bit Fractions	(HR) * (IR) ⇒ E : D (AM) + (E : D) ⇒ AM Qualified (IX) ⇒ IX Qualified (IY) ⇒ IY (HR) ⇒ IZ (M : M + 1)X ⇒ HR (M : M + 1)Y ⇒ IR
PSHMAC	Push MAC State	MAC Registers ⇒ Stack
PULMAC	Pull MAC State	Stack ⇒ MAC Registers
RMAC	Repeating Multiply and Accumulate Signed 16-Bit Fractions	Repeat until (E) < 0 (AM) + (H) * (I) ⇒ AM Qualified (IX) ⇒ IX; Qualified (IY) ⇒ IY; (M : M + 1)X ⇒ H; (M : M + 1)Y ⇒ I (E) – 1 ⇒ E
TDMSK	Transfer D to XMSK : YMSK	(D[15:8]) ⇒ XMSK (D[7:0]) ⇒ YMSK
TEDM	Transfer E and D to AM[31:0] Sign Extend AM	(D) ⇒ AM[15:0] (E) ⇒ AM[31:16] AM[32:35] = AM31
TEM	Transfer E to AM[31:16] Sign Extend AM Clear AM LSB	(E) ⇒ AM[31:16] $00 ⇒ AM[15:0] AM[32:35] = AM31
TMER	Transfer AM to E Rounded	Rounded (AM) ⇒ Temp If (SM • (EV + MV)) then Saturation ⇒ E else Temp[31:16] ⇒ E
TMET	Transfer AM to E Truncated	If (SM • (EV + MV)) then Saturation ⇒ E else AM[31:16] ⇒ E
TMXED	Transfer AM to IX : E : D	AM[35:32] ⇒ IX[3:0] AM35 ⇒ IX[15:4] AM[31:16] ⇒ E AM[15:0] ⇒ D

5.6.10 Stop and Wait Instructions

There are two instructions that put the CPU16 in an inactive state. Both require that either an interrupt or a reset exception occurs before normal execution of instructions resumes. However, each operates differently.

LPSTOP minimizes microcontroller power consumption. The CPU16 initiates a stop, but it and other controller modules are deactivated by the MCU system integration module (SIM). Reactivation is also handled by the SIM. The interrupt priority field from the CPU16 condition code register is copied into SIM control logic, then the system clock to the processor is stopped. When a reset or an interrupt of higher priority than the IP value occurs, the SIM activates the CPU16, and the appropriate exception processing sequence begins (refer to **SECTION 4 SYSTEM INTEGRATION MODULE** for more information).

WAI idles the CPU16, but does not affect operation of other MCU modules. The IP field is not copied to the SIM. System clocks continue to run. The processor waits until a reset or an interrupt of higher priority than the IP value occurs, then begins the appropriate exception processing sequence.

Because the system integration module does not restart the CPU16, interrupts are acknowledged more quickly following WAI than following LPSTOP.

To make certain that conditions for termination of LPSTOP and WAI are correct, interrupts are not recognized until after the instruction following ANDP, ORP, TAP, and TDP executes. This prevents interrupt exception processing during the period after the mask changes but before the following instruction executes. Refer to **5.12 Exceptions** and **SECTION 4 SYSTEM INTEGRATION MODULE** for more information concerning exceptions and interrupt exception processing.

Table 5–30. Stop and Wait Summary

Mnemonic	Function	Operation
LPSTOP	Low Power Stop	If $\overline{S}$ then STOP else NOP
WAI	Wait for Interrupt	WAIT

5.6.11 Background Mode and Null Operations

Background debugging mode (BDM) is a special CPU16 operating mode that is used for system development. Executing BGND when BDM is enabled puts the CPU16 in this mode. The logic state of certain SIM pins during system reset affects BDM operation. Refer to **SECTION 4 SYSTEM INTEGRATION MODULE** for more information.

Null operations are often used to replace other instructions during software debugging. Replacing conditional branch instructions with BRN, for instance, permits testing a decision-making routine without actually taking the branches.

Table 5–31. Background Mode and Null Operations

BGND	Enter Background Debug Mode	If BDM enabled enter BDM; else, illegal instruction
BRN	Branch Never	If 1 = 0, branch
LBRN	Long Branch Never	If 1 = 0, branch
NOP	Null operation	—

5.6.12 Instruction Set Summary

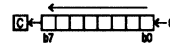
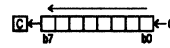
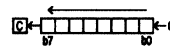
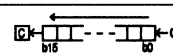
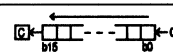
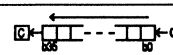
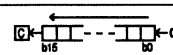
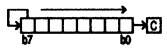
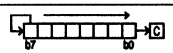
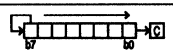
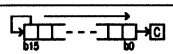
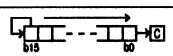
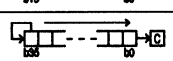
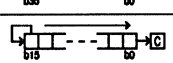
The following summary is a quick reference to the entire CPU16 instruction set. Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for detailed information on each instruction, assembler syntax, and condition code evaluation. A key to table nomenclature is provided on the last page of the table.

Instruction Set Summary

Mnemonic	Operation	Description	Address Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
ABA	Add B to A	(A) + (B) ⇒ A	INH	370B	—	2	—	—	Δ	—	Δ	Δ	Δ	Δ
ABX	Add B to X	(XK : IX) + (000 : B) ⇒ XK : IX	INH	374F	—	2	—	—	—	—	—	—	—	—
ABY	Add B to Y	(YK : IY) + (000 : B) ⇒ YK : IY	INH	375F	—	2	—	—	—	—	—	—	—	—
ABZ	Add B to Z	(ZK : IZ) + (000 : B) ⇒ ZK : IZ	INH	376F	—	2	—	—	—	—	—	—	—	—
ACE	Add E to AM[31:15]	(AM[31:15]) + (E) ⇒ AM	INH	3722	—	2	—	Δ	—	Δ	—	—	—	—
ACED	Add concatenated E and D to AM	(E : D) + (AM) ⇒ AM	INH	3723	—	4	—	Δ	—	Δ	—	—	—	—
ADCA	Add with Carry to A	(A) + (M) + C ⇒ A	IND8, X	43	ff	6	—	—	Δ	—	Δ	Δ	Δ	Δ
			IND8, Y	53	ff	6								
			IND8, Z	63	ff	6								
			IMM8	73	ii	2								
			IND16, X	1743	gggg	6								
			IND16, Y	1753	gggg	6								
			IND16, Z	1763	gggg	6								
			EXT	1773	hh ll	6								
			E, X	2743	—	6								
			E, Y	2753	—	6								
			E, Z	2763	—	6								
ADCB	Add with Carry to B	(B) + (M) + C ⇒ B	IND8, X	C3	ff	6	—	—	Δ	—	Δ	Δ	Δ	Δ
			IND8, Y	D3	ff	6								
			IND8, Z	E3	ff	6								
			IMM8	F3	ii	2								
			E, X	27C3	—	6								
			E, Y	27D3	—	6								
			E, Z	27E3	—	6								
			IND16, X	17C3	gggg	6								
			IND16, Y	17D3	gggg	6								
			IND16, Z	17E3	gggg	6								
			EXT	17F3	hh ll	6								
ADCD	Add with Carry to D	(D) + (M : M + 1) + C ⇒ D	IND8, X	83	ff	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	93	ff	6								
			IND8, Z	A3	ff	6								
			E, X	2783	—	6								
			E, Y	2793	—	6								
			E, Z	27A3	—	6								
			IMM16	37B3	jj kk	4								
			IND16, X	37C3	gggg	6								
			IND16, Y	37D3	gggg	6								
			IND16, Z	37E3	gggg	6								
			EXT	37F3	hh ll	6								
ADCE	Add with Carry to E	(E) + (M : M + 1) + C ⇒ E	IMM16	3733	jj kk	4	—	—	—	—	Δ	Δ	Δ	Δ
			IND16, X	3743	gggg	6								
			IND16, Y	3753	gggg	6								
			IND16, Z	3763	gggg	6								
			EXT	3773	hh ll	6								
ADDA	Add to A	(A) + (M) ⇒ A	IND8, X	41	ff	6	—	—	Δ	—	Δ	Δ	Δ	Δ
			IND8, Y	51	ff	6								
			IND8, Z	61	ff	6								
			IMM8	71	ii	2								
			E, X	2741	—	6								
			E, Y	2751	—	6								
			E, Z	2761	—	6								
			IND16, X	1741	gggg	6								
			IND16, Y	1751	gggg	6								
			IND16, Z	1761	gggg	6								
			EXT	1771	hh ll	6								

Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
ADDB	Add to B	(B) + (M) ⇒ B	IND8, X	C1	ii	6	—	—	Δ	—	Δ	Δ	Δ	Δ
			IND8, Y	D1	ff	6								
			IND8, Z	E1	ff	6								
			IMM8	F1	ff	2								
			E, X	27C1	—	6								
			E, Y	27D1	—	6								
			E, Z	27E1	—	6								
			IND16, X	17C1	gggg	6								
			IND16, Y	17D1	gggg	6								
			IND16, Z	17E1	gggg	6								
			EXT	17F1	hh ll	6								
ADDD	Add to D	(D) + (M : M + 1) ⇒ D	IND8, X	81	jj kk	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	91	ff	6								
			IND8, Z	A1	ff	6								
			IMM8	FC	ff	2								
			E, X	2781	—	6								
			E, Y	2791	—	6								
			E, Z	27A1	—	6								
			IMM16	37B1	ii	4								
			IND16, X	37C1	gggg	6								
			IND16, Y	37D1	gggg	6								
			IND16, Z	37E1	gggg	6								
			EXT	37F1	hh ll	6								
ADDE	Add to E	(E) + (M : M + 1) ⇒ E	IMM8	7C	ii	2	—	—	—	—	Δ	Δ	Δ	Δ
			IMM16	3731	jj kk	4								
			IND16, X	3741	gggg	6								
			IND16, Y	3751	gggg	6								
			IND16, Z	3761	gggg	6								
			EXT	3771	hh ll	6								
ADE	Add D to E	(E) + (D) ⇒ E	INH	2778	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ADX	Add D to X	(XK : IX) + («D) ⇒ XK : IX	INH	37CD	—	2	—	—	—	—	—	—	—	—
ADY	Add D to Y	(YK : IY) + («D) ⇒ YK : IY	INH	37DD	—	2	—	—	—	—	—	—	—	—
ADZ	Add D to Z	(ZK : IZ) + («D) ⇒ ZK : IZ	INH	37ED	—	2	—	—	—	—	—	—	—	—
AEX	Add E to X	(XK : IX) + («E) ⇒ XK : IX	INH	374D	—	2	—	—	—	—	—	—	—	—
AEY	Add E to Y	(YK : IY) + («E) ⇒ YK : IY	INH	375D	—	2	—	—	—	—	—	—	—	—
AEZ	Add E to Z	(ZK : IZ) + («E) ⇒ ZK : IZ	INH	376D	—	2	—	—	—	—	—	—	—	—
AIS	Add Immediate Data to SP	SK : SP + «IMM ⇒ SK : SP	IMM8	3F	ii	2	—	—	—	—	—	—	—	—
			IMM16	373F	jj kk	4								
AIX	Add Immediate Value to X	XK : IX + «IMM ⇒ XK : IX	IMM8	3C	ii	2	—	—	—	—	—	Δ	—	—
			IMM16	373C	jj kk	4								
AIY	Add Immediate Value to Y	YK : IY + «IMM ⇒ YK : IY	IMM8	3D	ii	2	—	—	—	—	—	Δ	—	—
			IMM16	373D	jj kk	4								
AIZ	Add Immediate Value to Z	ZK : IZ + «IMM ⇒ ZK : IZ	IMM8	3E	ii	2	—	—	—	—	—	Δ	—	—
			IMM16	373E	jj kk	4								
ANDA	AND A	(A) • (M) ⇒ A	IND8, X	46	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	56	ff	6								
			IND8, Z	66	ff	6								
			IMM8	76	ii	2								
			IND16, X	1746	gggg	6								
			IND16, Y	1756	gggg	6								
			IND16, Z	1766	gggg	6								
			EXT	1776	hh ll	6								
			E, X	2746	—	6								
			E, Y	2756	—	6								
			E, Z	2766	—	6								
ANDB	AND B	(B) • (M) ⇒ B	IND8, X	C6	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	D6	ff	6								
			IND8, Z	E6	ff	6								
			IMM8	F6	ii	2								
			IND16, X	17C6	gggg	6								
			IND16, Y	17D6	gggg	6								
			IND16, Z	17E6	gggg	6								
			EXT	17F6	hh ll	6								
			E, X	27C6	—	6								
			E, Y	27D6	—	6								
			E, Z	27E6	—	6								

Mnemonic	Operation	Description	Address Mode	Instruction Opcode	Instruction Operand	Cycles	Condition Codes S	MV	H	EV	N	Z	V	C
ANDD	AND D	(D) • (M : M + 1) ⇒ D	IND8, X	86	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	96	ff	6								
			IND8, Z	A6	ff	6								
			E, X	2786	—	6								
			E, Y	2796	—	6								
			E, Z	27A6	—	6								
			IMM16	37B6	jj kk	4								
			IND16, X	37C6	gggg	6								
			IND16, Y	37D6	gggg	6								
			IND16, Z	37E6	gggg	6								
			EXT	37F6	hh ll	6								
ANDE	AND E	(E) • (M : M + 1) ⇒ E	IMM16	3736	jj kk	4	—	—	—	—	Δ	Δ	0	—
			IND16, X	3746	gggg	6								
			IND16, Y	3756	gggg	6								
			IND16, Z	3766	gggg	6								
			EXT	3776	hh ll	6								
ANDP[1]	AND CCR	(CCR) • IMM16⇒ CCR	IMM16	373A	jj kk	4	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
ASL	Arithmetic Shift Left		IND8, X	04	ff	8	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	14	ff	8								
			IND8, Z	24	ff	8								
			IND16, X	1704	gggg	8								
			IND16, Y	1714	gggg	8								
			IND16, Z	1724	gggg	8								
			EXT	1734	hh ll	8								
ASLA	Arithmetic Shift Left A		INH	3704	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASLB	Arithmetic Shift Left B		INH	3714	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASLD	Arithmetic Shift Left D		INH	27F4	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASLE	Arithmetic Shift Left E		INH	2774	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASLM	Arithmetic Shift Left AM		INH	27B6	—	4	—	Δ	—	Δ	Δ	—	—	Δ
ASLW	Arithmetic Shift Left Word		IND16, X	2704	gggg	8	—	—	—	—	Δ	Δ	Δ	Δ
			IND16, Y	2714	gggg	8								
			IND16, Z	2724	gggg	8								
			EXT	2734	hh ll	8								
ASR	Arithmetic Shift Right		IND8, X	0D	ff	8	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	1D	ff	8								
			IND8, Z	2D	ff	8								
			IND16, X	170D	gggg	8								
			IND16, Y	171D	gggg	8								
			IND16, Z	172D	gggg	8								
			EXT	173D	hh ll	8								
ASRA	Arithmetic Shift Right A		INH	370D	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASRB	Arithmetic Shift Right B		INH	371D	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASRD	Arithmetic Shift Right D		INH	27FD	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASRE	Arithmetic Shift Right E		INH	277D	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASRM	Arithmetic Shift Right AM		INH	27BA	—	2	—	—	—	Δ	Δ	—	—	Δ
ASRW	Arithmetic Shift Right Word		IND16, X	270D	gggg	8	—	—	—	—	Δ	Δ	Δ	Δ
			IND16, Y	271D	gggg	8								
			IND16, Z	272D	gggg	8								
			EXT	273D	hh ll	8								
BCC	Branch if Carry Clear	If C = 0, branch	REL8	B4	rr	6, 2	—	—	—	—	—	—	—	—

Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
BCLR	Clear Bit(s)	(M) • ($\overline{\text{Mask}}$) ⇒ M	IND16, X	08	mm gggg	8	—	—	—	—	Δ	Δ	0	—
			IND16, Y	18	mm gggg	8								
			IND16, Z	28	mm gggg	8								
			EXT	38	mm hh ll	8								
			IND8, X	1708	mm ff	8								
			IND8, Y	1718	mm ff	8								
			IND8, Z	1728	mm ff	8								
BCLRW	Clear Bit(s) Word	(M : M + 1) • ($\overline{\text{Mask}}$) ⇒ M : M + 1	IND16, X	2708	mmmm gggg	10	—	—	—	—	Δ	Δ	0	—
			IND16, Y	2718	mmmm gggg	10								
			IND16, Z	2728	mmmm gggg	10								
			EXT	2738	mmmm hh ll	10								
BCS	Branch if Carry Set	If C = 1, branch	REL8	B5	rr	6, 2	—	—	—	—	—	—	—	—
BEQ	Branch if Equal	If Z = 1, branch	REL8	B7	rr	6, 2	—	—	—	—	—	—	—	—
BGE	Branch if Greater Than or Equal to Zero	If N ⊕ V = 0, branch	REL8	BC	rr	6, 2	—	—	—	—	—	—	—	—
BGND	Enter Background Debug Mode	If BDM enabled enter BDM; else, illegal instruction	INH	37A6	—	—	—	—	—	—	—	—	—	—
BGT	Branch if Greater Than Zero	If Z + (N ⊕ V) = 0, branch	REL8	BE	rr	6, 2	—	—	—	—	—	—	—	—
BHI	Branch if Higher	If C + Z = 0, branch	REL8	B2	rr	6, 2	—	—	—	—	—	—	—	—
BITA	Bit Test A	(A) • (M)	IND8, X	49	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	59	ff	6								
			IND8, Z	69	ff	6								
			IMM8	79	ii	2								
			IND16, X	1749	gggg	6								
			IND16, Y	1759	gggg	6								
			IND16, Z	1769	gggg	6								
			EXT	1779	hh ll	6								
			E, X	2749	—	6								
			E, Y	2759	—	6								
			E, Z	2769	—	6								
BITB	Bit Test B	(B) • (M)	IND8, X	C9	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	D9	ff	6								
			IND8, Z	E9	ff	6								
			IMM8	F9	ii	2								
			IND16, X	17C9	gggg	6								
			IND16, Y	17D9	gggg	6								
			IND16, Z	17E9	gggg	6								
			EXT	17F9	hh ll	6								
			E, X	27C9	—	6								
			E, Y	27D9	—	6								
			E, Z	27E9	—	6								
BLE	Branch if Less Than or Equal to Zero	If Z + (N ⊕ V) = 1, branch	REL8	BF	rr	6, 2	—	—	—	—	—	—	—	—
BLS	Branch if Lower or Same	If C + Z = 1, branch	REL8	B3	rr	6, 2	—	—	—	—	—	—	—	—
BLT	Branch if Less Than Zero	If N ⊕ V = 1, branch	REL8	BD	rr	6, 2	—	—	—	—	—	—	—	—
BMI	Branch if Minus	If N = 1, branch	REL8	BB	rr	6, 2	—	—	—	—	—	—	—	—
BNE	Branch if Not Equal	If Z = 0, branch	REL8	B6	rr	6, 2	—	—	—	—	—	—	—	—
BPL	Branch if Plus	If N = 0, branch	REL8	BA	rr	6, 2	—	—	—	—	—	—	—	—
BRA	Branch Always	If 1 = 1, branch	REL8	B0	rr	6	—	—	—	—	—	—	—	—
BRCLR	Branch if Bit(s) Clear	If (M) • (Mask) = 0, branch	IND8, X	CB	mm ff rr	12, 10	—	—	—	—	—	—	—	—
			IND8, Y	DB	mm ff rr	12, 10								
			IND8, Z	EB	mm ff rr	12, 10								
			IND16, X	0A	mm gggg rr	14, 10								
			IND16, Y	1A	mm gggg rr	14, 10								
			IND16, Z	2A	mm gggg rr	14, 10								
			EXT	3A	mm hh ll rr	14, 10								
BRN	Branch Never	If 1 = 0, branch	REL8	B1	rr	2	—	—	—	—	—	—	—	—

Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction Opcode	Instruction Operand	Instruction Cycles	Condition Codes S	MV	H	EV	N	Z	V	C
BRSET	Branch if Bit(s) Set	If (M) • (Mask) = 0, branch	IND8, X	8B	mm ff rr	12, 10	—	—	—	—	—	—	—	—
			IND8, Y	9B	mm ff rr	12, 10								
			IND8, Z	AB	mm ff rr	12, 10								
			IND16, X	0B	mm gggg rr	14, 10								
			IND16, Y	1B	mm gggg rr	14, 10								
			IND16, Z	2B	mm gggg rr	14, 10								
			EXT	3B	mm hh ll rr	14, 10								
BSET	Set Bit(s)	(M) • (Mask) ⇒ M	IND16, X	09	mm gggg	8	—	—	—	—	Δ	Δ	0	—
			IND16, Y	19	mm gggg	8								
			IND16, Z	29	mm gggg	8								
			EXT	39	mm hh ll	8								
			IND8, X	1709	mm ff	8								
			IND8, Y	1719	mm ff	8								
			IND8, Z	1729	mm ff	8								
BSETW	Set Bit(s) in Word	(M : M + 1) • (Mask) ⇒ M : M + 1	IND16, X	2709	mmmm gggg	10	—	—	—	—	Δ	Δ	0	—
			IND16, Y	2719	mmmm gggg	10								
			IND16, Z	2729	mmmm gggg	10								
			EXT	2739	mmmm hh ll	10								
BSR	Branch to Subroutine	(PK : PC) – 2 ⇒ PK : PC Push (PC) (SK : SP) – 2 ⇒ SK : SP Push (CCR) (SK : SP) – 2 ⇒ SK : SP (PK:PC) + Offset ⇒ PK:PC	REL8	36	rr	10	—	—	—	—	—	—	—	—
BVC	Branch if Overflow Clear	If V = 0, branch	REL8	B8	rr	6, 2	—	—	—	—	—	—	—	—
BVS	Branch if Overflow Set	If V = 1, branch	REL8	B9	rr	6, 2	—	—	—	—	—	—	—	—
CBA	Compare A to B	(A) – (B)	INH	371B	—	6, 2	—	—	—	—	Δ	Δ	Δ	Δ
CLR	Clear Memory	$00 ⇒ M	IND8, X	05	ff	4	—	—	—	—	0	1	0	0
			IND8, Y	15	ff	4								
			IND8, Z	25	ff	4								
			IND16, X	1705	gggg	6								
			IND16, Y	1715	gggg	6								
			IND16, Z	1725	gggg	6								
			EXT	1735	hh ll	6								
CLRA	Clear A	$00 ⇒ A	INH	3705	—	2	—	—	—	—	0	1	0	0
CLRB	Clear B	$00 ⇒ B	INH	3715	—	2	—	—	—	—	0	1	0	0
CLRD	Clear D	$0000 ⇒ D	INH	27F5	—	2	—	—	—	—	0	1	0	0
CLRE	Clear E	$0000 ⇒ E	INH	2775	—	2	—	—	—	—	0	1	0	0
CLRM	Clear AM	$000000000 ⇒ AM[32:0]	INH	27B7	—	2	—	0	—	0	—	—	—	—
CLRW	Clear Memory Word	$0000 ⇒ M : M + 1	IND16, X	2705	gggg	6	—	—	—	—	0	1	0	0
			IND16, Y	2715	gggg	6								
			IND16, Z	2725	gggg	6								
			EXT	2735	hh ll	6								
CMPA	Compare A to Memory	(A) – (M)	IND8, X	48	ff	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	58	ff	6								
			IND8, Z	68	ff	6								
			IMM8	78	ii	2								
			IND16, X	1748	gggg	6								
			IND16, Y	1758	gggg	6								
			IND16, Z	1768	gggg	6								
			EXT	1778	hh ll	6								
			E, X	2748	—	6								
			E, Y	2758	—	6								
			E, Z	2768	—	6								

Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction		Cycles	Condition Codes							
				Opcode	Operand		S	MV	H	EV	N	Z	V	C
CMPB	Compare B to Memory	(B) – (M)	IND8, X	C8	ff	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	D8	ff	6								
			IND8, Z	E8	ff	6								
			IMM8	F8	ii	2								
			IND16, X	17C8	gggg	6								
			IND16, Y	17D8	gggg	6								
			IND16, Z	17E8	gggg	6								
			EXT	17F8	hh ll	6								
			E, X	27C8	—	6								
			E, Y	27D8	—	6								
			E, Z	27E8	—	6								
COM	Ones Complement	$FF – (M) ⇒ M	IND8, X	00	ff	8	—	—	—	—	Δ	Δ	0	1
			IND8, Y	10	ff	8								
			IND8, Z	20	ff	8								
			IND16, X	1700	gggg	8								
			IND16, Y	1710	gggg	8								
			IND16, Z	1720	gggg	8								
			EXT	1730	hh ll	8								
COMA	Ones Complement A	$FF – (A) ⇒ A	INH	3700	—	2	—	—	—	—	Δ	Δ	0	1
COMB	Ones Complement B	$FF – (B) ⇒ B	INH	3710	—	2	—	—	—	—	Δ	Δ	0	1
COMD	Ones Complement D	$FFFF – (D) ⇒ D	INH	27F0	—	2	—	—	—	—	Δ	Δ	0	1
COME	Ones Complement E	$FFFF – (E) ⇒ E	INH	2770	—	2	—	—	—	—	Δ	Δ	0	1
COMW	Ones Complement Word	$FFFF – M : M + 1 ⇒ M : M + 1	IND16, X	2700	gggg	8	—	—	—	—	Δ	Δ	0	1
			IND16, Y	2710	gggg	8								
			IND16, Z	2720	gggg	8								
			EXT	2730	hh ll	8								
CPD	Compare D to Memory	(D) – (M : M + 1)	IND8, X	88	ff	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	98	ff	6								
			IND8, Z	A8	ff	6								
			E, X	2788	—	6								
			E, Y	2798	—	6								
			E, Z	27A8	—	6								
			IMM16	37B8	jj kk	4								
			IND16, X	37C8	gggg	6								
			IND16, Y	37D8	gggg	6								
			IND16, Z	37E8	gggg	6								
			EXT	37F8	hh ll	6								
CPE	Compare E to Memory	(E) – (M : M + 1)	IMM16	3738	hh ll	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND16, X	3748	gggg	6								
			IND16, Y	3758	gggg	6								
			IND16, Z	3768	gggg	6								
			EXT	3778	jj kk	6								
CPS	Compare SP to Memory	(SP) – (M : M + 1)	IND8, X	4F	ff	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	5F	ff	6								
			IND8, Z	6F	ff	6								
			IND16, X	174F	gggg	6								
			IND16, Y	175F	gggg	6								
			IND16, Z	176F	gggg	6								
			EXT	177F	hh ll	6								
			IMM16	377F	jj kk	4								
CPX	Compare IX to Memory	(IX) – (M : M + 1)	IND8, X	4C	ff	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	5C	ff	6								
			IND8, Z	6C	ff	6								
			IND16, X	174C	gggg	6								
			IND16, Y	175C	gggg	6								
			IND16, Z	176C	gggg	6								
			EXT	177C	hh ll	6								
			IMM16	377C	jj kk	4								
CPY	Compare IY to Memory	(IY) – (M : M + 1)	IND8, X	4D	ff	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	5D	ff	6								
			IND8, Z	6D	ff	6								
			IND16, X	174D	gggg	6								
			IND16, Y	175D	gggg	6								
			IND16, Z	176D	gggg	6								
			EXT	177D	hh ll	6								
			IMM16	377D	jj kk	4								

Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction Opcode	Instruction Operand	Instruction Cycles	Condition Codes S	MV	H	EV	N	Z	V	C
CPZ	Compare IZ to Memory	(IZ) – (M : M + 1)	IND8, X	4E	ff	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	5E	ff	6								
			IND8, Z	6E	ff	6								
			IND16, X	174E	gggg	6								
			IND16, Y	175E	gggg	6								
			IND16, Z	176E	gggg	6								
			EXT	177E	hh ll	6								
			IMM16	377E	jj kk	4								
DAA	Decimal Adjust A	$(A)_{10}$	INH	3721	1	2	—	—	—	—	Δ	Δ	Δ	Δ
DEC	Decrement Memory	(M) – $01 ⇒ M	IND8, X	01	ff	8	—	—	—	—	Δ	Δ	Δ	—
			IND8, Y	11	ff	8								
			IND8, Z	21	ff	8								
			IND16, X	1701	gggg	8								
			IND16, Y	1711	gggg	8								
			IND16, Z	1721	gggg	8								
			EXT	1731	hh ll	8								
DECA	Decrement A	(A) – $01 ⇒ A	INH	3701	—	2	—	—	—	—	Δ	Δ	Δ	—
DECB	Decrement B	(B) – $01 ⇒ B	INH	3711	—	2	—	—	—	—	Δ	Δ	Δ	—
DECW	Decrement Memory Word	(M : M + 1) – $0001 ⇒ M : M + 1	IND16, X	2701	gggg	8	—	—	—	—	Δ	Δ	Δ	—
			IND16, Y	2711	gggg	8								
			IND16, Z	2721	gggg	8								
			EXT	2731	hh ll	8								
EDIV	Extended Unsigned Divide	(E : D) / (IX) Quotient ⇒ IX Remainder ⇒ D	INH	3728	—	24	—	—	—	—	Δ	Δ	Δ	Δ
EDIVS	Extended Signed Divide	(E : D) / (IX) Quotient ⇒ IX Remainder ⇒ ACCD	INH	3729	—	38	—	—	—	—	Δ	Δ	Δ	Δ
EMUL	Extended Unsigned Multiply	(E) * (D) ⇒ E : D	INH	3725	—	10	—	—	—	—	Δ	Δ	—	Δ
EMULS	Extended Signed Multiply	(E) * (D) ⇒ E : D	INH	3726	—	8	—	—	—	—	Δ	Δ	—	Δ
EORA	Exclusive OR A	(A) ⊕ (M) ⇒ A	IND8, X	44	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	54	ff	6								
			IND8, Z	64	ff	6								
			IMM8	74	ii	2								
			IND16, X	1744	gggg	6								
			IND16, Y	1754	gggg	6								
			IND16, Z	1764	gggg	6								
			EXT	1774	hh ll	6								
			E, X	2744	—	6								
			E, Y	2754	—	6								
			E, Z	2764	—	6								
EORB	Exclusive OR B	(B) ⊕ (M) ⇒ B	IND8, X	C4	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	D4	ff	6								
			IND8, Z	E4	ff	6								
			IMM8	F4	ii	6								
			IND16, X	17C4	gggg	2								
			IND16, Y	17D4	gggg	6								
			IND16, Z	17E4	gggg	6								
			EXT	17F4	hh ll	6								
			E, X	27C4	—	6								
			E, Y	27D4	—	6								
			E, Z	27E4	—	6								
EORD	Exclusive OR D	(D) ⊕ (M : M + 1) ⇒ D	IND8, X	84	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	94	ff	6								
			IND8, Z	A4	ff	6								
			E, X	2784	—	6								
			E, Y	2794	—	6								
			E, Z	27A4	—	6								
			IMM16	37B4	hh ll	4								
			IND16, X	37C4	gggg	6								
			IND16, Y	37D4	gggg	6								
			IND16, Z	37E4	gggg	6								
			EXT	37F4	jj kk	6								
EORE	Exclusive OR E	(E) ⊕ (M : M + 1) ⇒ E	IMM16	3734	hh ll	4	—	—	—	—	Δ	Δ	0	—
			IND16, X	3744	gggg	6								
			IND16, Y	3754	gggg	6								
			IND16, Z	3764	gggg	6								
			EXT	3774	jj kk	6								

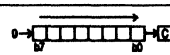
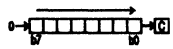
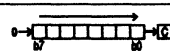
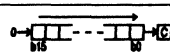
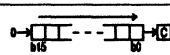
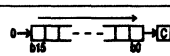
Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
FDIV	Fractional Divide	(D) / (IX) ⇒ IX Remainder ⇒ D	INH	372B	—	22	—	—	—	—	—	Δ	Δ	Δ
FMULS	Fractional Signed Multiply	(E) * (D) ⇒ E : D[31:1] 0 ⇒ D[0]	INH	3727	—	8	—	—	—	—	Δ	Δ	Δ	Δ
IDIV	Integer Divide	(D) / (IX) ⇒ IX; Remainder ⇒ D	INH	372A		22	—	—	—	—	—	Δ	0	Δ
INC	Increment Memory	(M) + $01 ⇒ M	IND8, X IND8, Y IND8, Z IND16, X IND16, Y IND16, Z EXT	03 13 23 1703 1713 1723 1733	ff ff ff gggg gggg gggg hh ll	2 2 2 2 2 2 2	—	—	—	—	Δ	Δ	Δ	—
INCA	Increment A	(A) + $01 ⇒ A	INH	3703	—	2	—	—	—	—	Δ	Δ	Δ	—
INCB	Increment B	(B) + $01 ⇒ B	INH	3713	—	2	—	—	—	—	Δ	Δ	Δ	—
INCW	Increment Memory Word	(M : M + 1) + $0001 ⇒ M : M + 1	IND16, X IND16, Y IND16, Z EXT	2703 2713 2723 2733	gggg gggg gggg hh ll	8 8 8 8	—	—	—	—	Δ	Δ	Δ	—
JMP	Jump	⟨ea⟩ ⇒ PK : PC	IND20, X IND20, Y IND20, Z EXT20	4B 5B 6B 7A	zg gggg zg gggg zg gggg zb hh ll	8 8 8 6	—	—	—	—	—	—	—	—
JSR	Jump to Subroutine	Push (PC) (SK : SP) – 2 ⇒ SK : SP Push (CCR) (SK : SP) – 2 ⇒ SK : SP ⟨ea⟩ ⇒ PK : PC	IND20, X IND20, Y IND20, Z EXT	89 99 A9 FA	zg gggg zg gggg zg gggg zb hh ll	12 12 12 10	—	—	—	—	—	—	—	—
LBCC	Long Branch if Carry Clear	If C = 0, branch	REL16	3784	rrrr	6, 4	—	—	—	—	—	—	—	—
LBCS	Long Branch if Carry Set	If C = 1, branch	REL16	3785	rrrr	6, 4	—	—	—	—	—	—	—	—
LBEQ	Long Branch if Equal	If Z = 1, branch	REL16	3787	rrrr	6, 4	—	—	—	—	—	—	—	—
LBEV	Long Branch if EV Set	If EV = 1, branch	REL16	3791	rrrr	6, 4	—	—	—	—	—	—	—	—
LBGE	Long Branch if Greater Than or Equal to Zero	If N ⊕ V = 0, branch	REL16	378C	rrrr	6, 4	—	—	—	—	—	—	—	—
LBGT	Long Branch if Greater Than Zero	If Z + (N ⊕ V) = 0, branch	REL16	378E	rrrr	6, 4	—	—	—	—	—	—	—	—
LBHI	Long Branch if Higher	If C + Z = 0, branch	REL16	3782	rrrr	6, 4	—	—	—	—	—	—	—	—
LBLE	Long Branch if Less Than or Equal to Zero	If Z + (N ⊕ V) = 1, branch	REL16	378F	rrrr	6, 4	—	—	—	—	—	—	—	—
LBLS	Long Branch if Lower or Same	If C + Z = 1, branch	REL16	3783	rrrr	6, 4	—	—	—	—	—	—	—	—
LBLT	Long Branch if Less Than Zero	If N ⊕ V = 1, branch	REL16	378D	rrrr	6, 4	—	—	—	—	—	—	—	—
LBMI	Long Branch if Minus	If N = 1, branch	REL16	378B	rrrr	6, 4	—	—	—	—	—	—	—	—
LBMV	Long Branch if MV Set	If MV = 1, branch	REL16	3790	rrrr	6, 4								
LBNE	Long Branch if Not Equal	If Z = 0, branch	REL16	3786	rrrr	6, 4	—	—	—	—	—	—	—	—
LBPL	Long Branch if Plus	If N = 0, branch	REL16	378A	rrrr	6, 4	—	—	—	—	—	—	—	—
LBRA	Long Branch Always	If 1 = 1, branch	REL16	3780	rrrr	6	—	—	—	—	—	—	—	—
LBRN	Long Branch Never	If 1 = 0, branch	REL16	3781	rrrr	6	—	—	—	—	—	—	—	—
LBSR	Long Branch to Subroutine	Push (PC) (SK : SP) – 2 ⇒ SK : SP Push (CCR) (SK : SP) – 2 ⇒ SK : SP (PK : PC) + Offset ⇒ PK : PC	REL16	27F9	rrrr	10	—	—	—	—	—	—	—	—
LBVC	Long Branch if Overflow Clear	If V = 0, branch	REL16	3788	rrrr	6, 4	—	—	—	—	—	—	—	—
LBVS	Long Branch if Overflow Set	If V = 1, branch	REL16	3789	rrrr	6, 4	—	—	—	—	—	—	—	—

Instruction Set Summary (Continued)

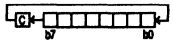
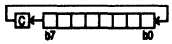
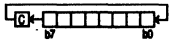
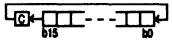
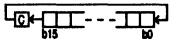
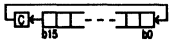
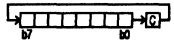
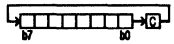
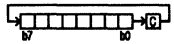
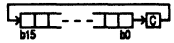
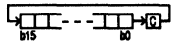
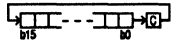
Mnemonic	Operation	Description	Address Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
LDAA	Load A	(M) ⇒ A	IND8, X	45	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	55	ff	6								
			IND8, Z	65	ff	6								
			IMM8	75	ii	2								
			IND16, X	1745	gggg	6								
			IND16, Y	1755	gggg	6								
			IND16, Z	1765	gggg	6								
			EXT	1775	hh ll	6								
			E, X	2745	—	6								
			E, Y	2755	—	6								
			E, Z	2765	—	6								
LDAB	Load B	(M) ⇒ B	IND8, X	C5	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	D5	ff	6								
			IND8, Z	E5	ff	6								
			IMM8	F5	ii	2								
			IND16, X	17C5	gggg	6								
			IND16, Y	17D5	gggg	6								
			IND16, Z	17E5	gggg	6								
			EXT	17F5	hh ll	6								
			E, X	27C5	—	6								
			E, Y	27D5	—	6								
			E, Z	27E5	—	6								
LDD	Load D	(M : M + 1) ⇒ D	IND8, X	85	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	95	ff	6								
			IND8, Z	A5	ff	6								
			E, X	2785	—	6								
			E, Y	2795	—	6								
			E, Z	27A5	—	6								
			IMM16	37B5	jj kk	4								
			IND16, X	37C5	gggg	6								
			IND16, Y	37D5	gggg	6								
			IND16, Z	37E5	gggg	6								
			EXT	37F5	hh ll	6								
LDE	Load E	(M : M + 1) ⇒ E	IMM16	3735	jj kk	4	—	—	—	—	Δ	Δ	0	—
			IND16, X	3745	gggg	6								
			IND16, Y	3755	gggg	6								
			IND16, Z	3765	gggg	6								
			EXT	3775	hh ll	6								
LDED	Load Concatenated E and D	(M : M + 1) ⇒ E (M + 2 : M + 3) ⇒ D	EXT	2771	hh ll	8	—	—	—	—	—	—	—	—
LDHI	Initialize H and I	(M : M + 1)$_X$ ⇒ H R (M : M + 1)$_Y$ ⇒ I R	INH	27B0	—	8	—	—	—	—	—	—	—	—
LDS	Load SP	(M : M + 1) ⇒ SP	IND8, X	CF	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	DF	ff	6								
			IND8, Z	EF	ff	6								
			IND16, X	17CF	gggg	6								
			IND16, Y	17DF	gggg	6								
			IND16, Z	17EF	gggg	6								
			EXT	17FF	hh ll	6								
			IMM16	37BF	jj kk	4								
LDX	Load IX	(M : M + 1) ⇒ IX	IND8, X	CC	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	DC	ff	6								
			IND8, Z	EC	ff	6								
			IND16, X	17CC	gggg	6								
			IND16, Y	17DC	gggg	6								
			IND16, Z	17EC	gggg	6								
			EXT	17FC	hh ll	6								
			IMM16	37BC	jj kk	4								
LDY	Load IY	(M : M + 1) ⇒ IY	IND8, X	CD	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	DD	ff	6								
			IND8, Z	ED	ff	6								
			IND16, X	17CD	gggg	6								
			IND16, Y	17DD	gggg	6								
			IND16, Z	17ED	gggg	6								
			EXT	17FD	hh ll	6								
			IMM16	37BD	jj kk	4								
LDZ	Load IZ	(M : M + 1) ⇒ IZ	IND8, X	CE	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	DE	ff	6								
			IND8, Z	EE	ff	6								
			IND16, X	17CE	gggg	6								
			IND16, Y	17DE	gggg	6								
			IND16, Z	17EE	gggg	6								
			EXT	17FE	hh ll	6								
			IMM16	37BE	jj kk	4								

Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction Opcode	Instruction Operand	Instruction Cycles	Condition Codes S	MV	H	EV	N	Z	V	C
LPSTOP	Low Power Stop	If S̄ then STOP else NOP	INH	27F1	—	4, 20	—	—	—	—	—	—	—	—
LSR	Logical Shift Right		IND8, X IND8, Y IND8, Z IND16, X IND16, Y IND16, Z EXT	0F 1F 2F 170F 171F 172F 173F	ff ff ff gggg gggg gggg hh ll	8 8 8 8 8 8 8	—	—	—	—	0	Δ	Δ	Δ
LSRA	Logical Shift Right A		INH	370F	—	2	—	—	—	—	0	Δ	Δ	Δ
LSRB	Logical Shift Right B		INH	371F	—	2	—	—	—	—	0	Δ	Δ	Δ
LSRD	Logical Shift Right D		INH	27FF	—	2	—	—	—	—	0	Δ	Δ	Δ
LSRE	Logical Shift Right E		INH	277F	—	2	—	—	—	—	0	Δ	Δ	Δ
LSRW	Logical Shift Right Word		IND16, X IND16, Y IND16, Z EXT	270F 271F 272F 273F	gggg gggg gggg hh ll	8 8 8 8	—	—	—	—	0	Δ	Δ	Δ
MAC	Multiply and Accumulate Signed 16-Bit Fractions	(HR) * (IR) ⇒ E : D (AM) + (E : D) ⇒ AM Qualified (IX) ⇒ IX Qualified (IY) ⇒ IY (HR) ⇒ IZ $(M : M + 1)_X$ ⇒ HR $(M : M + 1)_Y$ ⇒ IR	IMM8	7B	xoyo	12	—	Δ	—	Δ	—	—	Δ	—
MOVB	Move Byte	$(M_1) \Rightarrow M_2$	IXP to EXT EXT to IXP EXT to EXT	30 32 37FE	ff hh ll hh ll ff hh ll hh ll	8 8 10	—	—	—	—	Δ	Δ	0	—
MOVW	Move Word	$(M : M + 1_1) \Rightarrow M : M + 1_2$	IXP to EXT EXT to IXP EXT to EXT	31 33 37FF	ff hh ll hh ll ff hh ll hh ll	8 8 10	—	—	—	—	Δ	Δ	0	—
MUL	Multiply	(A) * (B) ⇒ D	INH	3724	—	10	—	—	—	—	—	—	—	Δ
NEG	Negate Memory	$00 – (M) ⇒ M	IND8, X IND8, Y IND8, Z IND16, X IND16, Y IND16, Z EXT	02 12 22 1702 1712 1722 1732	ff ff ff gggg gggg gggg hh ll	8 8 8 8 8 8 8	—	—	—	—	Δ	Δ	Δ	Δ
NEGA	Negate A	$00 – (A) ⇒ A	INH	3702	—	2	—	—	—	—	Δ	Δ	Δ	Δ
NEGB	Negate B	$00 – (B) ⇒ B	INH	3712	—	2	—	—	—	—	Δ	Δ	Δ	Δ
NEGD	Negate D	$0000 – (D) ⇒ D	INH	27F2	—	2	—	—	—	—	Δ	Δ	Δ	Δ
NEGE	Negate E	$0000 – (E) ⇒ E	INH	2772	—	2	—	—	—	—	Δ	Δ	Δ	Δ
NEGW	Negate Memory Word	$0000 – (M : M + 1) ⇒ M : M + 1	IND16, X IND16, Y IND16, Z EXT	2702 2712 2722 2732	gggg gggg gggg hh ll	8 8 8 8	—	—	—	—	Δ	Δ	Δ	Δ
NOP	Null Operation	—	INH	274C	—	2	—	—	—	—	—	—	—	—
ORAA	OR A	(A) + (M) ⇒ A	IND8, X IND8, Y IND8, Z IMM8 IND16, X IND16, Y IND16, Z EXT E, X E, Y E, Z	47 57 67 77 1747 1757 1767 1777 2747 2757 2767	ff ff ff ii gggg gggg gggg hh ll — — —	6 6 6 2 6 6 6 6 6 6 6	—	—	—	—	Δ	Δ	0	—

Mnemonic	Operation	Description	Address Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
ORAB	OR B	(B) + (M) ⇒ B	IND8, X IND8, Y IND8, Z IMM8 IND16, X IND16, Y IND16, Z EXT E, X E, Y E, Z	C7 D7 E7 F7 17C7 17D7 17E7 17F7 27C7 27D7 27E7	ff ff ff ii gggg gggg gggg hh ll — — —	6 6 6 2 6 6 6 6 6 6 6	—	—	—	—	Δ	Δ	0	—
ORD	OR D	(D) + (M : M + 1) ⇒ D	IND8, X IND8, Y IND8, Z E, X E, Y E, Z IMM16 IND16, X IND16, Y IND16, Z EXT	87 97 A7 2787 2797 27A7 37B7 37C7 37D7 37E7 37F7	ff ff ff — — — hh ll gggg gggg gggg jj kk	6 6 6 6 6 6 4 6 6 6 6	—	—	—	—	Δ	Δ	0	—
ORE	OR E	(E) + (M : M + 1) ⇒ E	IMM16 IND16, X IND16, Y IND16, Z EXT	3737 3747 3757 3767 3777	hh ll gggg gggg gggg jj kk	4 6 6 6 6	—	—	—	—	Δ	Δ	0	—
ORP [1]	OR Condition Code Register	(CCR) + IMM16 ⇒ CCR	IMM16	373B	jj kk	4	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
PSHA	Push A	(SK : SP) + 1 ⇒ SK : SP Push (A) (SK : SP) – 2 ⇒ SK : SP	INH	3708	—	4	—	—	—	—	—	—	—	—
PSHB	Push B	(SK : SP) + 1 ⇒ SK : SP Push (B) (SK : SP) – 2 ⇒ SK : SP	INH	3718	—	4	—	—	—	—	—	—	—	—
PSHM	Push Multiple Registers Mask bits: 0 = D 1 = E 2 = IX 3 = IY 4 = IZ 5 = K 6 = CCR 7 = (reserved)	For mask bits 0 to 7: If mask bit set Push register (SK : SP) – 2 ⇒ SK : SP	IMM8	34	ii	4 + 2N N = number of iterations	—	—	—	—	—	—	—	—
PSHMAC	Push MAC State	MAC Registers ⇒ Stack	INH	27B8	—	14	—	—	—	—	—	—	—	—
PULA	Pull A	(SK : SP) + 2 ⇒ SK : SP Pull (A) (SK : SP) – 1 ⇒ SK : SP	INH	3709	—	6	—	—	—	—	—	—	—	—
PULB	Pull B	(SK : SP) + 2 ⇒ SK : SP Pull (B) (SK : SP) – 1 ⇒ SK : SP	INH	3719	—	6	—	—	—	—	—	—	—	—
PULM [1]	Pull Multiple Registers Mask bits: 0 = CCR[15:4] 1 = K 2 = IZ 3 = IY 4 = IX 5 = E 6 = D 7 = (reserved)	For mask bits 0 to 7: If mask bit set (SK : SP) + 2 ⇒ SK : SP Pull register	IMM8	35	ii	2+2(N+1) N = number of iterations	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
PULMAC	Pull MAC State	Stack ⇒ MAC Registers	INH	27B9	—	16	—	—	—	—	—	—	—	—
RMAC	Repeating Multiply and Accumulate Signed 16-Bit Fractions	Repeat until (E) < 0 (AM) + (H) * (I) ⇒ AM Qualified (IX) ⇒ IX; Qualified (IY) ⇒ IY; (M : M + 1)X ⇒ H; (M : M + 1)Y ⇒ I (E) – 1 ⇒ E	IMM8	FB	xoyo	6 + 12 per iteration	—	Δ	—	Δ	—	—	—	—

Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction Opcode	Instruction Operand	Cycles	Condition Codes S	MV	H	EV	N	Z	V	C
ROL	Rotate Left		IND8, X IND8, Y IND8, Z IND16, X IND16, Y IND16, Z EXT	0C 1C 2C 170C 171C 172C 173C	ff ff ff gggg gggg gggg hh ll	8 8 8 8 8 8 8	—	—	—	—	Δ	Δ	Δ	Δ
ROLA	Rotate Left A		INH	370C	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ROLB	Rotate Left B		INH	371C	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ROLD	Rotate Left D		INH	27FC	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ROLE	Rotate Left E		INH	277C	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ROLW	Rotate Left Word		IND16, X IND16, Y IND16, Z EXT	270C 271C 272C 273C	gggg gggg gggg hh ll	8 8 8 8	—	—	—	—	Δ	Δ	Δ	Δ
ROR	Rotate Right		IND8, X IND8, Y IND8, Z IND16, X IND16, Y IND16, Z EXT	0E 1E 2E 170E 171E 172E 173E	ff ff ff gggg gggg gggg hh ll	8 8 8 8 8 8 8	—	—	—	—	Δ	Δ	Δ	Δ
RORA	Rotate Right A		INH	370E	—	2	—	—	—	—	Δ	Δ	Δ	Δ
RORB	Rotate Right B		INH	371E	—	2	—	—	—	—	Δ	Δ	Δ	Δ
RORD	Rotate Right D		INH	27FE	—	2	—	—	—	—	Δ	Δ	Δ	Δ
RORE	Rotate Right E		INH	277E	—	2	—	—	—	—	Δ	Δ	Δ	Δ
RORW	Rotate Right Word		IND16, X IND16, Y IND16, Z EXT	270E 271E 272E 273E	gggg gggg gggg hh ll	8 8 8 8	—	—	—	—	Δ	Δ	Δ	Δ
RTI[2]	Return from Interrupt	(SK : SP) + 2 ⇒ SK : SP Pull CCR (SK : SP) + 2 ⇒ SK : SP Pull PC (PK : PC) – 6 ⇒ PK : PC	INH	2777	—	12	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
RTS[3]	Return from Subroutine	(SK : SP) + 2 ⇒ SK : SP Pull PK (SK : SP) + 2 ⇒ SK : SP Pull PC (PK : PC) – 2 ⇒ PK : PC	INH	27F7	—	12	—	—	—	—	—	—	—	—
SBA	Subtract B from A	(A) – (B) ⇒ A	INH	370A	—	2	—	—	—	—	Δ	Δ	Δ	Δ
SBCA	Subtract with Carry from A	(A) – (M) – C ⇒ A	IND8, X IND8, Y IND8, Z IMM8 IND16, X IND16, Y IND16, Z EXT E, X E, Y E, Z	42 52 62 72 1742 1752 1762 1772 2742 2752 2762	ff ff ff ii gggg gggg gggg hh ll — — —	6 6 6 2 6 6 6 6 6 6 6	—	—	—	—	Δ	Δ	Δ	Δ

5

Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction Opcode	Instruction Operand	Instruction Cycles	Condition Codes S	MV	H	EV	N	Z	V	C
SBCB	Subtract with Carry from B	(B) – (M) – C ⇒ B	IND8, X	C2	ff	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	D2	ff	6								
			IND8, Z	E2	ff	6								
			IMM8	F2	ii	2								
			IND16, X	17C2	gggg	6								
			IND16, Y	17D2	gggg	6								
			IND16, Z	17E2	gggg	6								
			EXT	17F2	hh ll	6								
			E, X	27C2	—	6								
			E, Y	27D2	—	6								
			E, Z	27E2	—	6								
SBCD	Subtract with Carry from D	(D) – (M : M + 1) – C ⇒ D	IND8, X	82	ff	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	92	ff	6								
			IND8, Z	A2	ff	6								
			E, X	2782	—	6								
			E, Y	2792	—	6								
			E, Z	27A2	—	6								
			IMM16	37B2	hh ll	4								
			IND16, X	37C2	gggg	6								
			IND16, Y	37D2	gggg	6								
			IND16, Z	37E2	gggg	6								
			EXT	37F2	jj kk	6								
SBCE	Subtract with Carry from E	(E) – (M : M + 1) – C ⇒ E	IMM16	3732	hh ll	4	—	—	—	—	Δ	Δ	Δ	Δ
			IND16, X	3742	gggg	6								
			IND16, Y	3752	gggg	6								
			IND16, Z	3762	gggg	6								
			EXT	3772	jj kk	6								
SDE	Subtract D from E	(E) – (D)⇒ E	INH	2779	—	2	—	—	—	—	Δ	Δ	Δ	Δ
STAA	Store A	(A) ⇒ M	IND8, X	4A	ff	4	—	—	—	—	Δ	Δ	0	—
			IND8, Y	5A	ff	4								
			IND8, Z	6A	ff	4								
			IND16, X	174A	gggg	6								
			IND16, Y	175A	gggg	6								
			IND16, Z	176A	gggg	6								
			EXT	177A	hh ll	6								
			E, X	274A	—	4								
			E, Y	275A	—	4								
			E, Z	276A	—	4								
STAB	Store B	(B) ⇒ M	IND8, X	CA	ff	4	—	—	—	—	Δ	Δ	0	—
			IND8, Y	DA	ff	4								
			IND8, Z	EA	ff	4								
			IND16, X	17CA	gggg	6								
			IND16, Y	17DA	gggg	6								
			IND16, Z	17EA	gggg	6								
			EXT	17FA	hh ll	6								
			E, X	27CA	—	4								
			E, Y	27DA	—	4								
			E, Z	27EA	—	4								
STD	Store D	(D) ⇒ M : M + 1	IND8, X	8A	ff	4	—	—	—	—	Δ	Δ	0	—
			IND8, Y	9A	ff	4								
			IND8, Z	AA	ff	4								
			E, X	278A	—	6								
			E, Y	279A	—	6								
			E, Z	27AA	—	6								
			IND16, X	37CA	gggg	4								
			IND16, Y	37DA	gggg	4								
			IND16, Z	37EA	gggg	4								
			EXT	37FA	jj kk	4								
STE	Store E	(E) ⇒ M : M + 1	IND16, X	3742	gggg	6	—	—	—	—	Δ	Δ	0	—
			IND16, Y	3752	gggg	6								
			IND16, Z	3762	gggg	6								
			EXT	3772	jj kk	6								
STED	Store Concatenated D and E	(E) ⇒ M : M + 1 (D) ⇒ M + 2 : M + 3	EXT	2773	hh ll	8	—	—	—	—	—	—	—	—
STS	Store SP	(SP) ⇒ M : M + 1	IND8, X	8F	ff	4	—	—	—	—	Δ	Δ	0	—
			IND8, Y	9F	ff	4								
			IND8, Z	AF	ff	4								
			IND16, X	178F	gggg	6								
			IND16, Y	179F	gggg	6								
			IND16, Z	17AF	gggg	6								
			EXT	17BF	hh ll	6								

Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction Opcode	Instruction Operand	Instruction Cycles	Condition Codes S	MV	H	EV	N	Z	V	C
STX	Store IX	(IX) ⇒ M : M + 1	IND8, X	8C	ff	4	—	—	—	—	Δ	Δ	0	—
			IND8, Y	9C	ff	4								
			IND8, Z	AC	ff	4								
			IND16, X	178C	gggg	6								
			IND16, Y	179C	gggg	6								
			IND16, Z	17AC	gggg	6								
			EXT	17BC	hh ll	6								
STY	Store IY	(IY) ⇒ M : M + 1	IND8, X	8D	ff	4	—	—	—	—	Δ	Δ	0	—
			IND8, Y	9D	ff	4								
			IND8, Z	AD	ff	4								
			IND16, X	178D	gggg	6								
			IND16, Y	179D	gggg	6								
			IND16, Z	17AD	gggg	6								
			EXT	17BD	hh ll	6								
STZ	Store Z	(IZ) ⇒ M : M + 1	IND8, X	8E	ff	4	—	—	—	—	Δ	Δ	0	—
			IND8, Y	9E	ff	4								
			IND8, Z	AE	ff	4								
			IND16, X	178E	gggg	6								
			IND16, Y	179E	gggg	6								
			IND16, Z	17AE	gggg	6								
			EXT	17BE	hh ll	6								
SUBA	Subtract from A	(A) – (M) ⇒ A	IND8, X	40	ff	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	50	ff	6								
			IND8, Z	60	ff	6								
			IMM8	70	ii	2								
			IND16, X	1740	gggg	6								
			IND16, Y	1750	gggg	6								
			IND16, Z	1760	gggg	6								
			EXT	1770	hh ll	6								
			E, X	2740	—	6								
			E, Y	2750	—	6								
			E, Z	2760	—	6								
SUBB	Subtract from B	(B) – (M) ⇒ B	IND8, X	C0	ff	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	D0	ff	6								
			IND8, Z	E0	ff	6								
			IMM8	F0	ii	2								
			IND16, X	17C0	gggg	6								
			IND16, Y	17D0	gggg	6								
			IND16, Z	17E0	gggg	6								
			EXT	17F0	hh ll	6								
			E, X	27C0	—	6								
			E, Y	27D0	—	6								
			E, Z	27E0	—	6								
SUBD	Subtract from D	(D) – (M : M + 1) ⇒ D	IND8, X	80	ff	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	90	ff	6								
			IND8, Z	A0	ff	6								
			E, X	2780	—	6								
			E, Y	2790	—	6								
			E, Z	27A0	—	6								
			IMM16	37B0	hh ll	4								
			IND16, X	37C0	gggg	6								
			IND16, Y	37D0	gggg	6								
			IND16, Z	37E0	gggg	6								
			EXT	37F0	jj kk	6								
SUBE	Subtract from E	(E) – (M : M + 1) ⇒ E	IMM16	3730	hh ll	4	—	—	—	—	Δ	Δ	Δ	Δ
			IND16, X	3740	gggg	6								
			IND16, Y	3750	gggg	6								
			IND16, Z	3760	gggg	6								
			EXT	3770	jj kk	6								
SWI	Software Interrupt	(PK : PC) + 2 ⇒ PK : PC Push (PC) (SK : SP) – 2 ⇒ SK : SP Push (CCR) (SK : SP) – 2 ⇒ SK : SP $0 ⇒ PK SWI Vector ⇒ PC	INH	3720	—	16	—	—	—	—	—	—	—	—
SXT	Sign Extend B into A	If B7 = 1 then A = $FF else A = $00	INH	27F8	—	2	—	—	—	—	Δ	Δ	—	—
TAB	Transfer A to B	(A) ⇒ B	INH	3717	—	2	—	—	—	—	Δ	Δ	0	—
TAP	Transfer A to CCR	(A[7:0]) ⇒ CCR[15:8]	INH	37FD	—	4	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
TBA	Transfer B to A	(B) ⇒ A	INH	3707	—	2	—	—	—	—	Δ	Δ	0	—
TBEK	Transfer B to EK	(B) ⇒ EK	INH	27FA	—	2	—	—	—	—	—	—	—	—

Mnemonic	Operation	Description	Address Mode	Instruction Opcode	Instruction Operand	Instruction Cycles	Condition Codes S	MV	H	EV	N	Z	V	C
TBSK	Transfer B to SK	(B) ⇒ SK	INH	379F	—	2	—	—	—	—	—	—	—	—
TBXK	Transfer B to XK	(B) ⇒ XK	INH	379C	—	2	—	—	—	—	—	—	—	—
TBYK	Transfer B to YK	(B) ⇒ YK	INH	379D	—	2	—	—	—	—	—	—	—	—
TBZK	Transfer B to ZK	(B) ⇒ ZK	INH	379E	—	2	—	—	—	—	—	—	—	—
TDE	Transfer D to E	(D) ⇒ E	INH	277B	—	2	—	—	—	—	Δ	Δ	0	—
TDMSK	Transfer D to XMSK : YMSK	(D[15:8]) ⇒ X MASK (D[7:0]) ⇒ Y MASK	INH	372F	—	2	—	—	—	—	—	—	—	—
TDP[1]	Transfer D to CCR	(D) ⇒ CCR[15:4]	INH	372D	—	4	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
TED	Transfer E to D	(E) ⇒ D	INH	27FB	—	2	—	—	—	—	Δ	Δ	0	—
TEDM	Transfer E and D to AM[31:0] Sign Extend AM	(D) ⇒ AM[15:0] (E) ⇒ AM[31:16] AM[32:35] = AM31	INH	27B1	—	4	—	0	—	0	—	—	—	—
TEKB	Transfer EK to B	$0 ⇒ B[7:4] (EK) ⇒ B[3:0]	INH	27BB	—	2	—	—	—	—	—	—	—	—
TEM	Transfer E to AM[31:16] Sign Extend AM Clear AM LSB	(E) ⇒ AM[31:16] $00 ⇒ AM[15:0] AM[32:35] = AM31	INH	27B2	—	4	—	0	—	0	—	—	—	—
TMER	Transfer AM to E Rounded	Rounded (AM) ⇒ Temp If (SM • (EV + MV)) then Saturation ⇒ E else Temp[31:16] ⇒ E	INH	27B4	—	6	—	Δ	—	Δ	Δ	Δ	—	—
TMET	Transfer AM to E Truncated	If (SM • (EV + MV)) then Saturation ⇒ E else AM[31:16] ⇒ E	INH	27B5	—	2	—	—	—	—	Δ	Δ	—	—
TMXED	Transfer AM to IX : E : D	AM[35:32] ⇒ IX[3:0] AM35 ⇒ IX[15:4] AM[31:16] ⇒ E AM[15:0] ⇒ D	INH	27B3	—	6	—	—	—	—	—	—	—	—
TPA	Transfer CCR MSB to A	(CCR[15:8]) ⇒ A	INH	37FC	—	2	—	—	—	—	—	—	—	—
TPD	Transfer CCR to D	(CCR) ⇒ D	INH	372C	—	2	—	—	—	—	—	—	—	—
TSKB	Transfer SK to B	(SK) ⇒ B[3:0] $0 ⇒ B[7:4]	INH	37AF	—	2					—	—	—	—
TST	Test for Zero or Minus	(M) – $00	IND8, X IND8, Y IND8, Z IND16, X IND16, Y IND16, Z EXT	06 16 26 1706 1716 1726 1736	ff ff ff gggg gggg gggg hh ll	6 6 6 6 6 6 6	—	—	—	—	Δ	Δ	0	0
TSTA	Test A for Zero or Minus	(A) – $00	INH	3706	—	2	—	—	—	—	Δ	Δ	0	0
TSTB	Test B for Zero or Minus	(B) – $00	INH	3716	—	2	—	—	—	—	Δ	Δ	0	0
TSTD	Test D for Zero or Minus	(D) – $0000	INH	27F6	—	2	—	—	—	—	Δ	Δ	0	0
TSTE	Test E for Zero or Minus	(E) – $0000	INH	2776	—	2	—	—	—	—	Δ	Δ	0	0
TSTW	Test for Zero or Minus Word	(M : M + 1) – $0000	IND16, X IND16, Y IND16, Z EXT	2706 2716 2726 2736	gggg gggg gggg hh ll	6 6 6 6	—	—	—	—	Δ	Δ	0	0
TSX	Transfer SP to X	(SK : SP) + 2 ⇒ XK : IX	INH	274F		2	—	—	—	—	—	—	—	—
TSY	Transfer SP to Y	(SK : SP) + 2 ⇒ YK : IY	INH	275F		2	—	—	—	—	—	—	—	—
TSZ	Transfer SP to Z	(SK : SP) + 2 ⇒ ZK : IZ	INH	276F		2	—	—	—	—	—	—	—	—
TXKB	Transfer XK to B	$0 ⇒ B[7:4] (XK) ⇒ B[3:0]	INH	37AC	—	2	—	—	—	—	—	—	—	—
TXS	Transfer X to SP	(XK : IX) – 2 ⇒ SK : SP	INH	374E	—	2	—	—	—	—	—	—	—	—
TXY	Transfer X to Y	(XK : IX) ⇒ YK : IY	INH	275C	—	2	—	—	—	—	—	—	—	—
TXZ	Transfer X to Z	(XK : IX) ⇒ ZK : IZ	INH	276C	—	2	—	—	—	—	—	—	—	—
TYKB	Transfer YK to B	$0 ⇒ B[7:4] (YK) ⇒ B[3:0]	INH	37AD	—	2	—	—	—	—	—	—	—	—
TYS	Transfer Y to SP	(YK : IY) – 2 ⇒ SK : SP	INH	375E	—	2	—	—	—	—	—	—	—	—
TYX	Transfer Y to X	(YK : IY) ⇒ XK : IX	INH	274D	—	2	—	—	—	—	—	—	—	—

Instruction Set Summary (Concluded)

Mnemonic	Operation	Description	Address Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
TYZ	Transfer Y to Z	(YK : IY) ⇒ ZK : IZ	INH	276D	—	2	—	—	—	—	—	—	—	—
TZKB	Transfer ZK to B	$0 ⇒ B[7:4] (ZK) ⇒ B[3:0]	INH	37AE	—	2	—	—	—	—	—	—	—	—
TZS	Transfer Z to SP	(ZK : IZ) – 2 ⇒ SK : SP	INH	376E	—	2	—	—	—	—	—	—	—	—
TZX	Transfer Z to X	(ZK : IZ) ⇒ XK : IX	INH	274E	—	2	—	—	—	—	—	—	—	—
TZY	Transfer Z to Y	(ZK : IZ) ⇒ ZK : IY	INH	275E	—	2	—	—	—	—	—	—	—	—
WAI	Wait for Interrupt	WAIT	INH	27F3	—	8	—	—	—	—	—	—	—	—
XGAB	Exchange A with B	(A) ⇔ (B)	INH	371A	—	2	—	—	—	—	—	—	—	—
XGDE	Exchange D with E	(D) ⇔ (E)	INH	277A		2	—	—	—	—	—	—	—	—
XGDX	Exchange D with X	(D) ⇔ (IX)	INH	37CC		2	—	—	—	—	—	—	—	—
XGDY	Exchange D with Y	(D) ⇔ (IY)	INH	37DC		2	—	—	—	—	—	—	—	—
XGDZ	Exchange D with Z	(D) ⇔ (IZ)	INH	37EC		2	—	—	—	—	—	—	—	—
XGEX	Exchange E with X	(E) ⇔ (IX)	INH	374C		2	—	—	—	—	—	—	—	—
XGEY	Exchange E with Y	(E) ⇔ (IY)	INH	375C		2	—	—	—	—	—	—	—	—
XGEZ	Exchange E with Z	(E) ⇔ (IZ)	INH	376C		2	—	—	—	—	—	—	—	—

NOTES:

1. CCR[15:4] change according to results of operation — PK field is not affected.
2. CCR[15:0] change according to copy of CCR pulled from stack.
3. PK field changes according to state pulled from stack — the rest of the CCR is not affected.

5

Instruction Set Abbreviations and Symbols

A	—	Accumulator A
AM	—	Accumulator M
B	—	Accumulator B
CCR	—	Condition code register
D	—	Accumulator D
E	—	Accumulator E
EK	—	Extended addressing extension field
IR	—	MAC multiplicand register
HR	—	MAC multiplier register
IX	—	Index register X
IY	—	Index register Y
IZ	—	Index register Z
K	—	Address extension register
PC	—	Program counter
PK	—	Program counter extension field
SK	—	Stack pointer extension field
SL	—	Multiply and accumulate sign latch
SP	—	Stack pointer
XK	—	Index register X extension field
YK	—	Index register Y extension field
ZK	—	Index register Z extension field
XMSK	—	Modulo addressing index register X mask
YMSK	—	Modulo addressing index register Y mask
S	—	Stop disable control bit
MV	—	AM overflow indicator
H	—	Half carry indicator
EV	—	AM extended overflow indicator
N	—	Negative indicator
Z	—	Zero indicator
V	—	Twos complement overflow indicator
C	—	Carry/borrow indicator
IP	—	Interrupt priority field
SM	—	Saturation mode control bit
PK	—	Program counter extension field
—	—	Bit not affected
Δ	—	Bit changes as specified
0	—	Bit cleared
1	—	Bit set
M	—	Memory location used in operation
R	—	Result of operation
S	—	Source data

X	—	Register used in operation
M	—	Address of one memory byte
M +1	—	Address of byte at M + $0001
M : M + 1	—	Address of one memory word
(...)X	—	Contents of address pointed to by IX
(...)Y	—	Contents of address pointed to by IY
(...)Z	—	Contents of address pointed to by IZ
E, X	—	IX with E offset
E, Y	—	IY with E offset
E, Z	—	IZ with E offset
EXT	—	Extended
EXT20	—	20-bit extended
IMM8	—	8-bit immediate
IMM16	—	16-bit immediate
IND8, X	—	IX with unsigned 8-bit offset
IND8, Y	—	IY with unsigned 8-bit offset
IND8, Z	—	IZ with unsigned 8-bit offset
IND16, X	—	IX with signed 16-bit offset
IND16, Y	—	IY with signed 16-bit offset
IND16, Z	—	IZ with signed 16-bit offset
IND20, X	—	IX with signed 20-bit offset
IND20, Y	—	IY with signed 20-bit offset
IND20, Z	—	IZ with signed 20-bit offset
INH	—	Inherent
IXP	—	Post-modified indexed
REL8	—	8-bit relative
REL16	—	16-bit relative
b	—	4-bit address extension
ff	—	8-bit unsigned offset
gggg	—	16-bit signed offset
hh	—	High byte of 16-bit extended address
ii	—	8-bit immediate data
jj	—	High byte of 16-bit immediate data
kk	—	Low byte of 16-bit immediate data
ll	—	Low byte of 16-bit extended address
mm	—	8-bit mask
mmmm	—	16-bit mask
rr	—	8-bit unsigned relative offset
rrrr	—	16-bit signed relative offset
xo	—	MAC index register X offset
yo	—	MAC index register Y offset
z	—	4-bit zero extension

+	—	Addition
-	—	Subtraction or negation (2'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

5.7 Comparison of CPU16 and MC68HC11 Instruction Sets

Most HC11 instructions are a source-code compatible subset of the CPU16 instruction set. However, certain HC11 instructions have been replaced by functionally equivalent HC16 instructions, and some CPU16 instructions with the same mnemonics as HC11 instructions operate differently.

Table 5–32 shows HC11 instructions that have either been replaced by CPU16 instructions or that operate differently in the CPU16. Replacement instructions are not identical to HC11 instructions — HC11 code must be altered to establish proper preconditions.

All CPU16 instruction execution times differ from those of the HC11. Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for details.

Table 5–32. HC16 Implementation of HC11 Instructions

HC11 Instruction	HC16 Implementation
BHS	BCC Only
BLO	BCS Only
BSR	Generates a different stack frame
CLC	Replaced by ANDP
CLI	Replaced by ANDP
CLV	Replaced by ANDP
DES	Replaced by AIS
DEX	Replaced by AIX
DEY	Replaced by AIY
INS	Replaced by AIS
INX	Replaced by AIX
INY	Replaced by AIY
JMP	IND8 addressing modes replaced by IND20 and EXT modes
JSR	IND8 addressing modes replaced by IND20 and EXT modes Generates a different stack frame
LSL, LSLD	Use ASL instructions*
PSHX	Replaced by PSHM
PSHY	Replaced by PSHM
PULX	Replaced by PULM
PULY	Replaced by PULM
RTI	Reloads PC and CCR only
RTS	Uses two-word stack frame
SEC	Replaced by ORP
SEI	Replaced by ORP
SEV	Replaced by ORP
STOP	Replaced by LPSTOP
TAP	CPU16 CCR bits differ from HC11 CPU16 interrupt priority scheme differs from HC11
TPA	CPU16 CCR bits differ from HC11 CPU16 interrupt priority scheme differs from HC11
TSX	Adds 2 to SK : SP before transfer to XK : IX
TSY	Adds 2 to SK : SP before transfer to YK : IY
TXS	Subtracts 2 from XK : IX before transfer to SK : SP
TXY	Transfers XK field to YK field
TYS	Subtracts 2 from YK : IY before transfer to SK : SP
TYX	Transfers YK field to XK field
WAI	Waits indefinitely for interrupt or reset Generates a different stack frame

*Motorola assemblers automatically translate ASL mnemonics

5.8 Instruction Format

CPU16 instructions consist of an 8-bit opcode, which may be preceded by an 8-bit prebyte and followed by one or more operands.

Opcodes are mapped in four 256-instruction pages. Page 0 opcodes stand alone, but Page 1, 2, and 3 opcodes are pointed to by a prebyte code on Page 0. The prebytes are $17 (Page 1), $27 (Page 2), and $37 (Page 3).

Operands can be 4 bits, 8 bits or 16 bits in length. However, because the CPU16 fetches 16-bit instruction words from even byte boundaries, each instruction must contain an even number of bytes.

Operands are organized as bytes, words, or a combination of bytes and words. Operands of 4-bits are either zero-extended to 8 bits, or packed two to a byte. The largest instructions are six bytes in length. Size, order, and function of operands are evaluated when an instruction is decoded.

A Page 0 opcode and an 8-bit operand can be fetched simultaneously. Instructions that use 8-bit indexed, immediate, and relative addressing modes have this form — code written with these instructions is very compact.

Table 5–33 shows basic CPU16 instruction formats.

Table 5–33. Basic Instruction Formats

8-Bit Opcode with 8-Bit Operand

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

8-Bit Opcode with 4-Bit Index Extensions

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

8-Bit Opcode, Argument(s)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Opcode								Operand							
Operand(s)															
Operand(s)															

8-Bit Opcode with 8-Bit Prebyte, No Argument

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

8-Bit Opcode with 8-Bit Prebyte, Argument(s)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Prebyte								Opcode							
Operand(s)															
Operand(s)															

8-Bit Opcode with 20-Bit Argument

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

5.9 Execution Model

This description builds up a conceptual model of the mechanism the CPU16 uses to fetch and execute instructions. The functional divisions in the model do not necessarily correspond to physical subunits of the microprocessor.

As shown in Figure 5–4, there are three functional blocks involved in fetching, decoding, and executing instructions. These are the microsequencer, the instruction pipeline, and the execution unit. These elements function concurrently — at any given time, all three may be active.

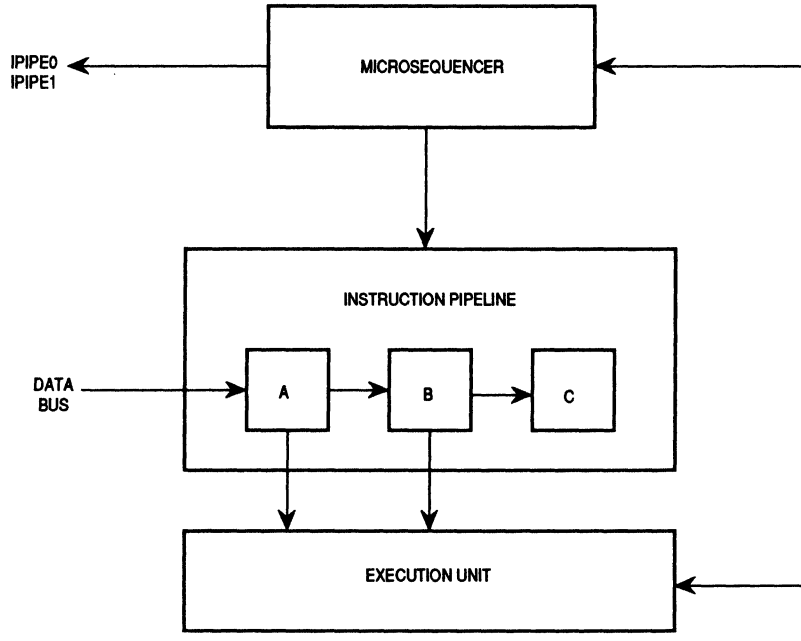


Figure 5–4. Instruction Execution Model

5.9.1 Microsequencer

The microsequencer controls the order in which instructions are fetched, advanced through the pipeline, and executed. It increments the program counter and generates multiplexed external tracking signals IPIPE0 and IPIPE1 from internal signals that control execution sequence.

5.9.2 Instruction Pipeline

The pipeline is a three stage FIFO that holds instructions while they are decoded and executed. Depending upon instruction size, as many as three instructions can be in the pipeline at one time (single-word instructions, one held in Stage C, one being executed in Stage B, and one latched in Stage A).

5.9.3 Execution Unit

The execution unit evaluates opcodes, interfaces with the microsequencer to advance instructions through the pipeline, and performs instruction operations.

5.10 Execution Process

Fetched opcodes are latched into Stage A, then advanced to Stage B. Opcodes are evaluated in Stage B. The execution unit can access operands in either Stage A or Stage B (Stage B accesses are limited to 8-bit operands). When execution is complete, opcodes are moved from Stage B to Stage C, where they remain until the next instruction is complete.

A prefetch mechanism in the microsequencer reads instruction words from memory and increments the program counter. When instruction execution begins, the program counter points to an address six bytes after the address of the first word of the instruction being executed.

The number of machine cycles necessary to complete an execution sequence varies according to the complexity of the instruction. Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for details.

5

5.10.1 Changes in Program Flow

When program flow changes, instructions are fetched from a new address. Before execution can begin at the new address, instructions and operands from the previous instruction stream must be removed from the pipeline. If a change in flow is temporary, a return address must be stored, so that execution of the original instruction stream can resume after the change in flow.

At the time an instruction that causes a change in program flow executes, PK : PC point to the address of the first word of the instruction + $0006. During execution of the instruction, PK : PC is loaded with the address of the first instruction word in the new instruction stream. However, Stages A and B still contain words from the old instruction stream. Extra processing steps must be performed prior to execution from the new instruction stream.

5.11 Instruction Timing

CPU16 instruction execution time has three components:

Bus cycles required to prefetch the next instruction
Bus cycles required for operand accesses
Time required for internal operations

A bus cycle requires a minimum of two system clock periods. If the access time of a memory device is greater than two clock periods, bus cycles will be longer. However, all bus cycles must be an integer number of clock periods. CPU16 internal operations are always an integer multiple of two clock periods.

Dynamic bus sizing affects bus cycle time. The system integration module manages all accesses. Refer to **SECTION 4 SYSTEM INTEGRATION MODULE** for more information.

The CPU16 does not execute more than one instruction at a time. The total time required to execute a particular instruction stream can be calculated by summing the individual execution times of each instruction in the stream.

Total execution time is calculated using the expression

$$(CL_T) = (CL_P) + (CL_O) + (CL_I)$$

Where:

(CL_T) = Total clock periods per instruction
(CL_I) = Clock periods used for internal operation
(CL_P) = Clock periods used for program access
(CL_O) = Clock periods used for operand access

A detailed discussion of instruction timing parameters is beyond the scope of this manual. Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for more information on this topic.

5.12 Exceptions

An exception is an event that preempts normal instruction process. 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. Exception processing includes all operations required to transfer control to a handler routine, but does not include execution of the handler routine itself. Keep the distinction between exception processing and execution of an exception handler in mind while reading this section.

5.12.1 Exception Vectors

An exception vector is the address of a routine that handles an exception. Exception vectors are contained in a data structure called the exception vector table, which is located in the first 512 bytes of Bank 0.

All vectors except the reset vector consist of one word and reside in data space. The reset vector consists of four words that reside in program space. (Refer to **SECTION 4 SYSTEM INTEGRATION MODULE** for information concerning address space types and the function code outputs.) There are 52 predefined or reserved vectors, and 200 user-defined vectors.

Each vector is assigned an 8-bit number. Vector numbers for some exceptions are generated by external devices; others are supplied by the processor. There is a direct mapping of vector number to vector table address. The processor left shifts the vector number one place (multiplies by two) to convert it to an address.

Table 5–34. Exception Vector Table

Vector Number	Vector Address	Address Space	Type of Exception
0	0000	P	RESET — Initial ZK, SK, and PK
	0002	P	RESET — Initial PC
	0004	P	RESET — Initial SP
	0006	P	RESET — Initial IZ (Direct Page)
4	0008	D	BKPT (Breakpoint)
5	000A	D	BERR (Bus Error)
6	000C	D	SWI (Software Interrupt)
7	000E	D	Illegal Instruction
8	0010	D	Division by Zero
9–E	0012–001C	D	Unassigned, Reserved
F	001E	D	Uninitialized Interrupt
10	0020	D	Unassigned, Reserved
11	0022	D	Level 1 Interrupt Autovector
12	0024	D	Level 2 Interrupt Autovector
13	0026	D	Level 3 Interrupt Autovector
14	0028	D	Level 4 Interrupt Autovector
15	002A	D	Level 5 Interrupt Autovector
16	002C	D	Level 6 Interrupt Autovector
17	002E	D	Level 7 Interrupt Autovector
18	0030	D	Spurious Interrupt
19–37	0032–006E	D	Unassigned, Reserved
38–FF	0070–01FE	D	User-Defined Interrupts

5.12.2 Exception Stack Frame

During exception processing, the contents of the program counter and condition code register are stacked at a location pointed to by SK : SP. Unless it is altered during exception processing, the stacked PK : PC value is the address of the next instruction in the current instruction stream, plus $0006. Figure 5–5 shows the exception stack frame.

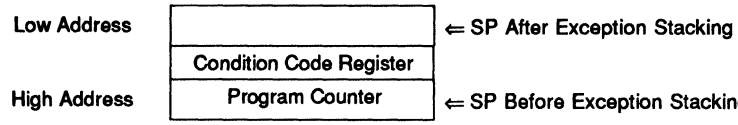


Figure 5–5. Exception Stack Frame Format

5.12.3 Exception Processing Sequence

Exception processing is performed in four distinct phases.

A. Priority of all pending exceptions is evaluated, and the highest priority exception is processed first.
B. Processor state is stacked, then the CCR PK extension field is cleared.
C. An exception vector number is acquired and converted to a vector address.
D. The content of the vector address is loaded into the PC, and the processor jumps to the exception handler routine.

There are variations within each phase for differing types of exceptions. However, all vectors but the reset vectors contain 16-bit addresses, and the PK field is cleared. Exception handlers must be located within Bank 0 or vectors must point to a jump table.

5.12.4 Types of Exceptions

Exceptions can be either internally or externally generated. External exceptions, which are defined as asynchronous, include interrupts, bus errors (BERR), breakpoints (BKPT), and resets (RESET). Internal exceptions, which are defined as synchronous, include the software interrupt (SWI) instruction, the background (BGND) instruction, illegal instruction exceptions, and the divide-by-zero exception.

5.12.4.1 Asynchronous Exceptions

Asynchronous exceptions occur without reference to CPU16 or IMB clocks, but exception processing is synchronized. For all asynchronous exceptions but RESET, exception processing begins at the first instruction boundary following recognition of an exception. Refer to **SECTION 4 SYSTEM INTEGRATION MODULE** for more information concerning asynchronous exceptions.

Because of pipelining, the stacked return PK : PC value for all asynchronous exceptions, other than RESET, is equal to the address of the next instruction in the current instruction stream plus $0006. The RTI instruction, which must terminate all exception handler routines, subtracts $0006 from the stacked value in order to resume execution of the interrupted instruction stream.

5.12.4.2 Synchronous Exceptions

Synchronous exception processing is part of an instruction definition. Exception processing for synchronous exceptions will always be completed, and the first instruction of the handler routine will always be executed, before interrupts are detected.

Because of pipelining, the value of PK : PC at the time a synchronous exception executes is equal to the address of the instruction that causes the exception plus $0006. Since RTI always subtracts $0006 upon return, the stacked PK : PC must be adjusted by the instruction that caused the exception so that execution will resume with the following instruction. For this reason $0002 is added to the PK : PC value before it is stacked.

5.12.5 Multiple Exceptions

Each exception has a hardware priority based upon its relative importance to system operation. Asynchronous exceptions have higher priorities than synchronous exceptions. Exception processing for multiple exceptions is done by priority, from lowest to highest. Priority governs the order in which exception processing occurs, not the order in which exception handlers are executed.

Unless BERR, BKPT, or RESET occur during exception processing, the first instruction of all exception handler routines is guaranteed to execute before another exception is processed. Since interrupt exceptions have higher priority than synchronous exceptions, this means that the first instruction in an interrupt handler will be executed before other interrupts are sensed.

RESET, BERR, and BKPT exceptions that occur during exception processing of a previous exception will be processed before the first instruction of that exception's handler routine. The converse is not true — if an interrupt occurs during BERR exception processing, for example, the first instruction of the BERR handler will be executed before interrupts are sensed. This permits the exception handler to mask interrupts during execution.

Refer to **SECTION 4 SYSTEM INTEGRATION MODULE** for detailed information concerning interrupts and system reset. Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for information concerning processing of specific exceptions.

5.12.6 RTI Instruction

The return-from-interrupt instruction (RTI) must be the last instruction in all exception handlers except the RESET handler. RTI pulls the exception stack frame that was pushed onto the system stack during exception processing, and restores processor state. Normal program flow resumes at the address of the instruction that follows the last instruction executed before exception processing began.

RTI is not used in the RESET handler because RESET initializes the stack pointer and does not create a stack frame.

5.13 Development Support

The CPU16 incorporates powerful tools for tracking program execution and for system debugging. These tools are deterministic opcode tracking, breakpoint exceptions, and background debugging mode. Judicious use of CPU16 capabilities permits in-circuit emulation and system debugging using a bus state analyzer, a simple serial interface, and a terminal.

5.13.1 Deterministic Opcode Tracking

The CPU16 has two multiplexed outputs, IPIPE0 and IPIPE1, that enable external hardware to monitor the instruction pipeline during normal program execution. The signals IPIPE0 and IPIPE1 can be demultiplexed into six pipeline state signals that allow a state analyzer to synchronize with instruction stream activity.

5.13.1.1 IPIPE0/IPIPE1 Multiplexing

Six types of information are required to track pipeline activity. To generate the six state signals, eight pipeline states are encoded and multiplexed into IPIPE0 and IPIPE1. The multiplexed signals have two phases. State signals are active low. Table 5–35 shows the encoding scheme.

Table 5–35. IPIPE0/IPIPE1 Encoding

Phase	IPIPE1 State	IPIPE0 State	State Signal Name
1	0	0	START & FETCH
	0	1	FETCH
	1	0	START
	1	1	NULL
2	0	0	INVALID
	0	1	ADVANCE
	1	0	EXCEPTION
	1	1	NULL

IPIPE0 and IPIPE1 are timed so that a logic analyzer can capture all six pipeline state signals and address, data, or control bus state in any single bus cycle. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for specifications.

State signals can be latched asynchronously on the falling and rising edges of either address strobe ($\overline{AS}$) or data strobe ($\overline{DS}$). They can also be latched synchronously using the microcontroller CLKOUT signal. Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for detailed information concerning state signals and state signal demux logic.

5.13.1.2 Combining Opcode Tracking with Other Capabilities

Pipeline state signals are useful during normal instruction execution and execution of exception handlers. The signals provide a complete model of the pipeline up to the point a breakpoint is acknowledged.

Breakpoints are acknowledged after an instruction has executed, when it is in pipeline Stage C. A breakpoint can initiate either exception processing or background debugging mode. IPIPE0/IPIPE1 are not usable when the CPU16 is in background debugging mode.

5.13.2 Breakpoints

Breakpoints are set by internal assertion of the IMB $\overline{BKPT}$ signal or by external assertion of the microcontroller $\overline{BKPT}$ pin. In the MC68HC16Z1, no internal module can assert the IMB $\overline{BKPT}$ signal. The CPU16 supports breakpoints on any memory access. Acknowledged breakpoints can initiate either exception processing or background debugging mode. After BDM has been enabled, the CPU16 will enter BDM when either $\overline{BKPT}$ input is asserted.

If $\overline{BKPT}$ assertion is synchronized with an instruction prefetch, the instruction is tagged with the breakpoint when it enters the pipeline, and the breakpoint occurs after the instruction executes.

If $\overline{BKPT}$ assertion is synchronized with an operand fetch, breakpoint processing occurs at the end of the instruction during which $\overline{BKPT}$ is latched.

Breakpoints on instructions that are flushed from the pipeline before execution are not acknowledged, but operand breakpoints are always acknowledged. There is no breakpoint acknowledge bus cycle when BDM is entered. Refer to **SECTION 4 SYSTEM INTEGRATION MODULE** for more information concerning breakpoints.

5.13.3 Opcode Tracking and Breakpoints

Breakpoints are acknowledged after a tagged instruction has executed, when it is copied from pipeline Stage B to Stage C. Stage C contains the opcode of the previous instruction when execution of the current instruction begins.

When an instruction is tagged, IPIPE0/IPIPE1 reflect the start of execution and the appropriate number of pipeline advances and operand fetches before the breakpoint is acknowledged. If background debugging mode is enabled, these signals model the pipeline before BDM is entered.

5.13.4 Background Debugging Mode

Microprocessor debugging programs are generally implemented in external software. CPU16 BDM provides a debugger implemented in CPU microcode.

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

BDM is an alternate CPU16 operating mode. While the CPU16 is in BDM, normal instruction execution is suspended, and special microcode performs debugging functions under external control. While in BDM, the CPU16 ceases to fetch instructions via the parallel bus and communicates with the development system via a dedicated serial interface.

5.13.4.1 Enabling BDM

The CPU16 samples the internal and external $\overline{\text{BKPT}}$ signals during reset to determine whether to enable BDM. If either $\overline{\text{BKPT}}$ input is at logic level zero when sampled, an internal BDM enabled flag is set.

BDM operation is enabled when $\overline{\text{BKPT}}$ is asserted at the rising edge of the $\overline{\text{RESET}}$ signal. BDM remains enabled until the next system reset. If $\overline{\text{BKPT}}$ is at logic level one on the trailing edge of $\overline{\text{RESET}}$, BDM is disabled. $\overline{\text{BKPT}}$ is relatched on each rising transition of $\overline{\text{RESET}}$. $\overline{\text{BKPT}}$ is synchronized internally, and must be asserted for at least two clock cycles prior to negation of $\overline{\text{RESET}}$.

5.13.4.2 BDM Sources

When BDM is enabled, external breakpoint hardware, internal IMB module breakpoints, and the BGND instruction can cause the CPU16 to enter BDM. If BDM is not enabled when a breakpoint occurs, a breakpoint exception is processed.

5.13.4.2.1 $\overline{\text{BKPT}}$ Signal

If enabled, BDM is initiated when assertion of $\overline{\text{BKPT}}$ is acknowledged. There is no breakpoint acknowledge bus cycle when BDM is entered. Refer to **SECTION 4 SYSTEM INTEGRATION MODULE** for more information concerning breakpoint acknowledge cycles. For timing specifications refer to **APPENDIX A ELECTRICAL CHARACTERISTICS**.

5.13.4.2.2 BGND Instruction

If BDM has been enabled, executing BGND will cause the CPU16 to suspend normal operation and enter BDM. If BDM has not been correctly enabled, an illegal instruction exception is generated.

5.13.4.3 Entering BDM

5

When the processor detects a breakpoint or decodes a BGND instruction, it suspends instruction execution and asserts the FREEZE signal. Once FREEZE has been asserted, the CPU enables the serial communication hardware and awaits a command.

Assertion of FREEZE causes opcode tracking signals IPIPE0 and IPIPE1 to change definition and become serial communication signals DSO and DSI. FREEZE is asserted at the next instruction boundary after $\overline{\text{BKPT}}$ is asserted. IPIPE0 and IPIPE1 change function before an EXCEPTION signal can be generated. The development system must use FREEZE assertion as an indication that BDM has been entered. When BDM is exited, FREEZE is negated prior to initiation of normal bus cycles — IPIPE0 and IPIPE1 will be valid when normal instruction prefetch begins.

5.13.4.4 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. The BDM command set is summarized in Table 5–36. Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for a BDM command glossary.

Table 5–36. Command Summary

Command	Mnemonic	Description
Read Registers from Mask	RREGM	Read contents of registers specified by command word register mask
Write Registers from Mask	WREGM	Write to registers specified by command word register mask
Read MAC Registers	RDMAC	Read contents of entire Multiply and Accumulate register set
Write MAC Registers	WRMAC	Write to entire Multiply and Accumulate register set
Read PC and SP	RPCSP	Read contents of program counter and stack pointer
Write PC and SP	WPCSP	Write to program counter and stack pointer
Read Data Memory	RDMEM	Read byte from specified 20-bit address in data space
Write Data Memory	WDMEM	Write byte to specified 20-bit address in data space
Read Program Memory	RPMEM	Read word from specified 20-bit address in program space
Write Program Memory	WPMEM	Write word to specified 20-bit address in program space
Execute from current PK : PC	GO	Instruction pipeline flushed and refilled; instructions executed from current PC – $0006
Null Operation	NOP	Null command — performs no operation

5.13.4.5 Returning from BDM

BDM is terminated when a resume execution (GO) command is received. GO refills the instruction pipeline from address (PK : PC – $0006). FREEZE is negated prior to the first prefetch. Upon negation of FREEZE, the serial subsystem is disabled, and the DSO/DSI signals revert to IPIPE0/IPIPE1 functionality.

5.13.4.6 BDM Serial Interface

The serial interface uses a synchronous protocol similar to that of the Motorola Serial Peripheral Interface (SPI). Figure 5–6 is a development system serial logic diagram.

The development system serves as the master of the serial link, and is responsible for the generation of serial interface clock signal DSCLK.

Serial clock frequency range is from DC to one-half the CPU16 clock frequency. If DSCLK is derived from the CPU16 system clock, development system serial logic can be synchronized with the target processor.

The serial interface operates in full-duplex mode. Data transfers 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 status of CPU-generated messages.

Command and data transfers initiated by the development system must clear bit 16. All commands that return a result return 16 bits of data plus one status bit.

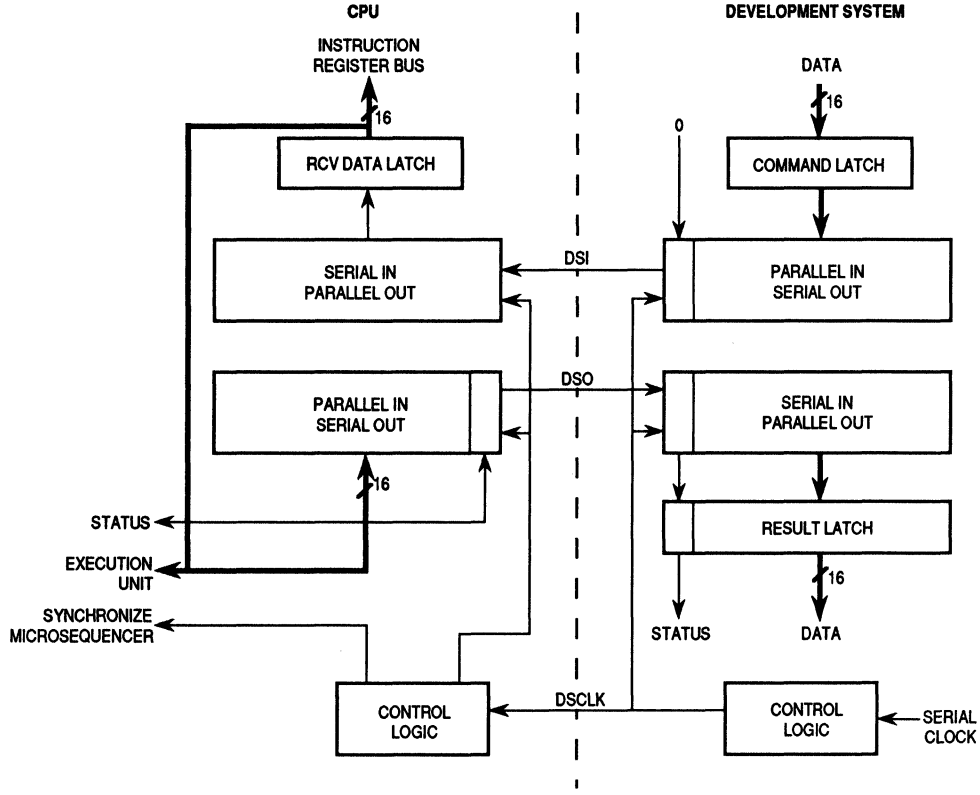


Figure 5–6. BDM Serial I/O Block Diagram

5.14 Digital Signal Processing

The CPU16 performs low-frequency digital signal processing algorithms in real time. The most common DSP operation in embedded control applications is filtering, but the CPU16 can perform several other useful DSP functions. These include autocorrelation (detecting a periodic signal in the presence of noise), cross-correlation (determining the presence of a defined periodic signal), and closed-loop control routines (selective filtration in a feedback path).

Although derivation of DSP algorithms is often a complex mathematic task, the algorithms themselves typically consist of a series of multiply and accumulate (MAC) operations. The CPU16 contains a dedicated set of registers that are used to perform MAC operations. These are collectively called the MAC unit.

DSP operations generally require a large number of MAC iterations. The CPU16 instruction set includes instructions that perform MAC setup and repetitive MAC operations. Other instructions, such as 32-bit load and store instructions, can also be used in DSP routines.

Many DSP algorithms require extensive data address manipulation. To increase throughput, the CPU16 performs effective address calculations and data prefetches during MAC operations. In addition, the MAC unit provides modulo addressing to efficiently implement circular DSP buffers.

Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for detailed information concerning the MAC unit and execution of DSP instructions.

SECTION 6
ANALOG-TO-DIGITAL CONVERTER

The analog-to-digital converter module (ADC) is a unipolar, successive-approximation converter with eight modes of operation. It has selectable 8- or 10-bit resolution. Monotonicity is guaranteed in both modes. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for ADC timing and electrical specifications. This section is an overview of ADC function. Refer to the *ADC Reference Manual* (ADCRM/AD) for a comprehensive discussion of ADC capabilities.

6.1 Overview

A bus interface unit (ABIU) handles communication between the ADC and other microcontroller modules, and supplies IMB timing signals to the ADC. Special operating modes and test functions are controlled by a module configuration register (ADCMCR) and a factory test register (ADCTST).

ADC module conversion functions can be grouped into three basic subsystems: an analog front end, a digital control section, and result storage. Figure 6–1 is a functional block diagram of the ADC module.

In addition to use as multiplexer inputs, the eight analog inputs can be used as a general-purpose digital input port (Port AD), provided signals are within logic level specification. A port data register (PADR) is used to access input data.

6.2 External Connections

The ADC uses 12 pins on the MC68HC16Z1 package. Eight pins are analog inputs (which can also be used as digital inputs), two pins are analog reference connections, and two pins are analog supply connections.

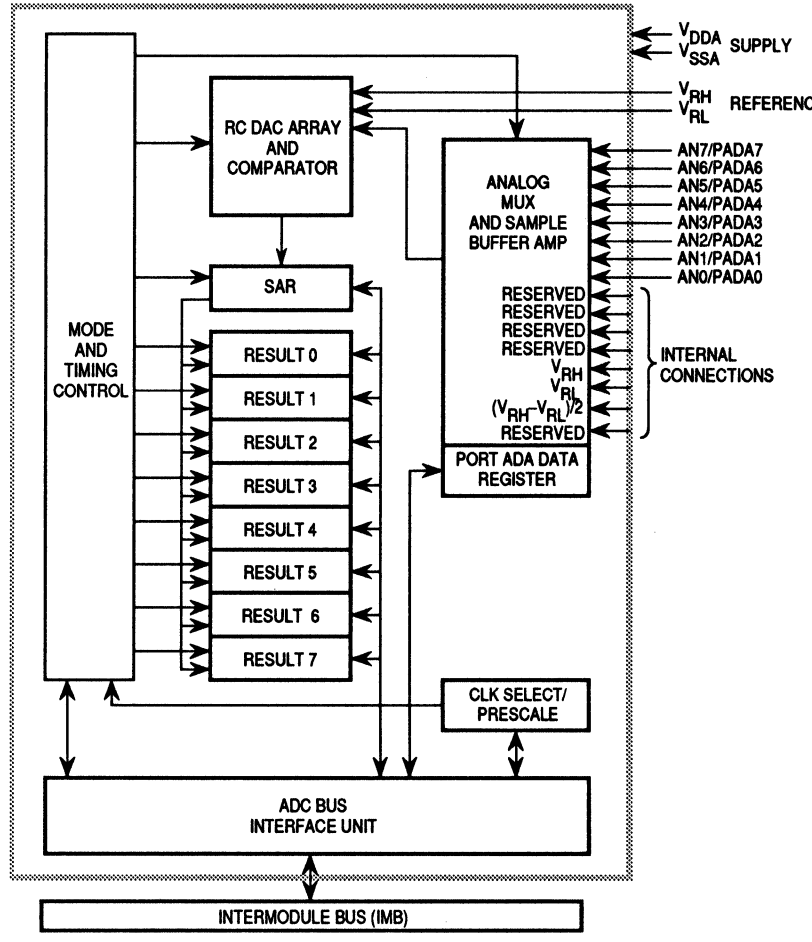


Figure 6–1. ADC Block Diagram

6.2.1 Analog Input Pins

Each of the eight analog input pins (AN[7:0]) is connected to a multiplexer in the ADC. The multiplexer selects an analog input for conversion to digital data.

Analog input pins can also be read as digital inputs, provided the applied voltage meet V_{IH} and V_{IL} specification. When used as digital inputs, the pins are organized into an 8-bit port (Port ADA), and referred to as ADA[7:0]. Digital input data is accessed via a port data register (PADR). There is no data direction register because port pins are used only for input.

6.2.2 Analog Reference Pins

Separate high (V_{RH}) and low (V_{RL}) analog reference voltages are connected to the analog reference pins. The pins permit connection of regulated and filtered supplies that allow the ADC to achieve its highest degree of accuracy.

6.2.3 Analog Supply Pins

Pins V_{DDA} and V_{SSA} supply power to analog circuitry associated with the RC DAC. Other circuitry in the ADC is powered from the digital power bus (pins V_{DDI} and V_{SSI}). Dedicated analog power supplies are necessary to isolate sensitive ADC circuitry from noise on the digital power bus.

6.3 Programmer's Model

The ADC module is mapped into 32 words of address space (refer to Table 6–1). Five words are control/status registers, one word is digital port data, and 24 words provide access to the results of AD conversion (eight addresses for each type of converted data). Two words are reserved for expansion.

The ADC module base address is determined by the value of the MM bit in the system integration module configuration register (SIMCR). The base address is normally $FFF700 in the MC68HC16Z1.

Internally, the ADC has both a differential data bus and a buffered IMB data bus. Registers not directly associated with conversion functions, such as the module configuration register, the module test register, and the port data register, reside on the buffered bus, while conversion registers and result registers reside on the differential bus.

Registers that reside on the buffered bus are updated immediately when written. However, writes to ADC control registers abort any conversion in progress.

6.4 ADC Bus Interface Unit

The ADC is designed to act as a slave device on the intermodule bus. The bus interface unit (ABIU) provides IMB bus cycle termination and synchronizes internal ADC signals with IMB signals. The ABIU also manages data bus routing to accommodate the three conversion data formats, and controls the interface to the module differential data bus.

6.5 Special Operating Modes

Low-power stop mode and freeze mode are ADC operating modes associated with assertion of IMB signals by other microcontroller modules or by external sources. These modes are controlled by the values of bits in the ADC module configuration register (ADCMCR).

6.5.1 Low-Power Stop Mode

When the STOP bit in ADCMCR is set, the IMB clock signal to the ADC is disabled. This places the module in an idle state, and power consumption is minimized. The bus interface unit does not shut down and ADC registers are still accessible. If a conversion is in progress when STOP is set, it is aborted.

STOP is set during system reset, and must be cleared before the ADC can be used. Because analog circuit bias currents are turned off during low-power stop, the ADC requires recovery time after STOP is cleared.

Execution of the CPU16 LPSTOP command places the entire modular microcontroller in low-power stop mode. Refer to **SECTION 4 SYSTEM INTEGRATION MODULE** and **SECTION 5 CENTRAL PROCESSING UNIT** for more information regarding low-power stop operation.

6.5.2 Freeze Mode

When the CPU16 in the modular microcontroller enters background debugging mode, the FREEZE signal is asserted. The ADC can respond to internal assertion of FREEZE in one of three different ways — it can ignore FREEZE assertion, finish the current conversion and then freeze, or freeze immediately.

Type of response is determined by the value of the FRZ[1:0] field in the module configuration register (refer to Table 6–2).

Table 6–1.
FRZ Field Selection

FRZ	Response
00	Ignore FREEZE
01	Reserved
10	Finish conversion, then freeze
11	Freeze immediately

When the ADC freezes, the ADC clock stops and all sequential activity ceases. Contents of control and status registers remain valid while frozen. When the FREEZE signal is negated, ADC activity resumes.

If the ADC freezes during a conversion, activity resumes with the next step in the conversion sequence. However, capacitors in the analog conversion circuitry discharge while the ADC is frozen — the conversion will be inaccurate.

Refer to **SECTION 5 CENTRAL PROCESSING UNIT** for more information on background debugging mode.

6.6 Analog Subsystem

The analog subsystem consists of a multiplexer, sample capacitors, a buffer amplifier, an RC DAC array, and a high-gain comparator. Comparator output is used to sequence the successive approximation register (SAR). The interface between the comparator and the SAR is the boundary between ADC analog and digital subsystems.

6.6.1 Multiplexer

The multiplexer selects one of 16 sources for conversion. Eight sources are internal and eight are external. Multiplexer operation is controlled by channel selection field CD:CA in register ADCTL1 (refer to Table 6–3). The multiplexer contains positive and negative stress protection circuitry. This circuitry prevents voltages on other input channels from affecting the current conversion.

Table 6–2.
Multiplexer Channels

[CD:CA] Value	Input Source
%0000	AN0
%0001	AN1
%0010	AN2
%0011	AN3
%0100	AN4
%0101	AN5
%0110	AN6
%0111	AN7
%1000	RESERVED
%1001	RESERVED
%1010	RESERVED
%1011	RESERVED
%1100	V_{RH}
%1101	V_{RL}
%1110	$(V_{RH} - V_{RL}) / 2$
%1111	TEST/RESERVED

6.6.2 Sample Capacitors and Buffer Amplifier

Each of the input channels has its own sample capacitor. All channels share a single buffer amplifier. After a channel is selected, for the first two ADC clock cycles of a sampling period, multiplexer output is connected to the input of the sample buffer amplifier via the sample capacitor. The sample amplifier buffers the input channel from the relatively large capacitance of the RC DAC array.

During the second two clock cycles of a sampling period, the sample capacitor is disconnected from the multiplexer, and the sample buffer amplifier charges the RC DAC array with the value stored in the sample capacitor.

During the third portion of a sampling period, both sample capacitor and buffer amplifier are bypassed, and multiplexer input charges the DAC array directly. The length of this third portion of a sampling period is determined by the value of the STS field in ADCTL0.

6.6.3 RC DAC Array

The RC DAC array consists of binary-weighted capacitors and a resistor-divider chain. The array performs two functions: it acts as a sample hold circuit during conversion, and it provides each successive digital-to-analog comparison voltage to the comparator. Conversion begins with MSB comparison and ends with LSB comparison. Array switching is controlled by the digital subsystem.

6.6.4 Comparator

The comparator indicates whether each approximation output from the RC DAC array during resolution is higher or lower than the sampled input voltage. Comparator output is fed to the digital control logic, which sets or clears each bit in the successive approximation register in sequence, MSB first.

6.7 Digital Control Subsystem

The digital control subsystem includes control and status registers, clock and prescaler control logic, channel and reference select logic, conversion sequence control logic, and the successive approximation register.

The subsystem controls the multiplexer and the output of the RC array during sample and conversion periods, stores the results of comparison in the successive-approximation register, then transfers results to the result registers.

6.7.1 Control/Status Registers

There are two control registers (ADCTL0, ADCTL1) and one status register (ADSTAT). ADCTL0 controls conversion resolution, sample time, and clock/prescaler value. ADCTL1 controls analog input selection, conversion mode, and initiation of conversion. A write to ADCTL0 aborts the current conversion sequence and halts the ADC. Conversion must be restarted by writing to ADCTL1. A write to ADCTL1 aborts the current conversion sequence and starts a new sequence with parameters altered by the write. ADSTAT shows conversion sequence status, conversion channel status, and conversion completion status.

The following paragraphs are a general discussion of control function. **APPENDIX D REGISTER SUMMARY** shows the ADC address map and discusses register bits and fields.

6.7.2 Clock and Prescaler Control

The ADC clock is derived from the system clock by a programmable prescaler. ADC clock period is determined by the value of the PRS field in ADCTL0.

The prescaler has two stages. The first stage is a 5-bit modulus counter. It divides the system clock by any value from 2 to 32 (PRS[4:0] = %00001 to %11111). The second stage is a divide-by-two circuit. Table 6–4 shows prescaler output values.

Table 6–3.
Prescaler Output

PRS[4:0]	ADC Clk
%00000	RESERVED
%00001	Sys Clk/4
%00010	Sys Clk/6
...	...
%11101	Sys Clk/60
%11110	Sys Clk/62
%11111	Sys Clk/64

ADC clock speed must be between 0.5 MHz and 2.1 MHz. The reset value of the PRS field is %00011, which divides a nominal 16.78-MHz system clock by eight, yielding maximum ADC clock frequency. There are a minimum of four IMB clock cycles for each ADC clock cycle.

6

6.7.3 Sample Time

The first two portions of all sample periods require four ADC clock cycles. During the third portion of a sample period, the selected channel is connected directly to the RC DAC array for a specified number of clock cycles. The value of the STS field in ADCTL0 determines the number of cycles (refer to Table 6–5). The number of clock cycles required for a sample period is the value specified by STS plus four. Sample time is determined by PRS value.

Table 6–4.
STS Field Selection

STS[1:0]	Sample Time
00	4 A/D Clock Periods
01	8 A/D Clock Periods
10	16 A/D Clock Periods
11	32 A/D Clock Periods

6.7.4 Resolution

ADC resolution can be either eight or ten bits. Resolution is determined by the state of the RES10 bit in ADCTL0. Both 8-bit and 10-bit conversion results are automatically aligned in the result registers.

6.7.5 Conversion Control Logic

Analog-to-digital conversions are performed in sequences. Sequences are initiated by any write to ADCTL1. If a conversion sequence is already in progress, a write to either control register will abort it and reset the SCF and CCF flags in the A/D status register. There are eight conversion modes. Conversion mode is determined by ADCTL1 control bits. Each conversion mode affects the bits in status register ADSTAT differently. Result storage differs from mode to mode.

6.7.5.1 Conversion Parameters

The following conversion parameters are controlled by bits in ADCTL1.

Conversion channel — the value of the channel selection field ([CA:CD]) in ADCTL1 determines which multiplexer inputs are used in a conversion sequence. There are 16 possible inputs. Eight inputs are external pins (AN[7:0]), and eight are internal.

Length of sequence — A conversion sequence consists of either four or eight conversions. The number of conversions in a sequence is determined by the state of the S8CM bit in ADCTL1.

Single or continuous conversion — Conversion can be limited to a single sequence or a sequence can be performed continuously. The state of the SCAN bit in ADCTL1 determines whether single or continuous conversion is performed.

Single or multiple channel conversion — Conversion sequence(s) can be run on a single channel or on a block of four or eight channels. Channel conversion is controlled by the state of the MULT bit in ADCTL1.

6.7.5.2 Conversion Modes

Conversion modes are defined by the state of the SCAN, MULT, and S8CM bits in ADCTL1. Table 6–6 shows mode numbering.

Table 6–5.
ADC Conversion Modes

SCAN	MULT	S8CM	Mode
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Mode 0 — A single 4-conversion sequence is performed on a single input channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT3). The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the conversion sequence is complete.

Mode 1 — A single 8-conversion sequence is performed on a single input channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT7). The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the conversion sequence is complete.

Mode 2 — A single conversion is performed on each of four sequential input channels, starting with the channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT3). The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the last conversion is complete.

Mode 3 — A single conversion is performed on each of eight sequential input channels, starting with the channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT7). The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the last conversion is complete.

Mode 4 — Continuous 4-conversion sequences are performed on a single input channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT3). Previous results are overwritten when a sequence repeats. The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the first 4-conversion sequence is complete.

Mode 5 — Continuous 8-conversion sequences are performed on a single input channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT7). Previous results are overwritten when a sequence repeats. The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the first 8-conversion sequence is complete.

Mode 6 — Continuous conversions are performed on each of four sequential input channels, starting with the channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT3). The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the first 4-conversion sequence is complete.

Mode 7 — Continuous conversions are performed on each of eight sequential input channels, starting with the channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT7). The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the first 8-conversion sequence is complete.

Table 6–7 summarizes ADC operation when MULT is cleared (single channel modes). Table 6–8 summarizes ADC operation when MULT is set (multi-channel modes). Number of conversions per channel is determined by SCAN. Channel numbers are given in order of conversion.

Table 6–6. Single-Channel Conversions

S8CM	CD	CC	CB	CA	Input	Result Register
0	0	0	0	0	AN0	RSLT[0:3]
0	0	0	0	1	AN1	RSLT[0:3]
0	0	0	1	0	AN2	RSLT[0:3]
0	0	0	1	1	AN3	RSLT[0:3]
0	0	1	0	0	AN4	RSLT[0:3]
0	0	1	0	1	AN5	RSLT[0:3]
0	0	1	1	0	AN6	RSLT[0:3]
0	0	1	1	1	AN7	RSLT[0:3]
0	1	0	0	0	RESERVED	RSLT[0:3]
0	1	0	0	1	RESERVED	RSLT[0:3]
0	1	0	1	0	RESERVED	RSLT[0:3]
0	1	0	1	1	RESERVED	RSLT[0:3]
0	1	1	0	0	V_{RH}	RSLT[0:3]
0	1	1	0	1	V_{RL}	RSLT[0:3]
0	1	1	1	0	$(V_{RH} - V_{RL}) / 2$	RSLT[0:3]
0	1	1	1	1	TEST/RESERVED	RSLT[0:3]
1	0	0	0	0	AN0	RSLT[0:7]
1	0	0	0	1	AN1	RSLT[0:7]
1	0	0	1	0	AN2	RSLT[0:7]
1	0	0	1	1	AN3	RSLT[0:7]
1	0	1	0	0	AN4	RSLT[0:7]
1	0	1	0	1	AN5	RSLT[0:7]
1	0	1	1	0	AN6	RSLT[0:7]
1	0	1	1	1	AN7	RSLT[0:7]
1	1	0	0	0	RESERVED	RSLT[0:7]
1	1	0	0	1	RESERVED	RSLT[0:7]
1	1	0	1	0	RESERVED	RSLT[0:7]
1	1	0	1	1	RESERVED	RSLT[0:7]
1	1	1	0	0	V_{RH}	RSLT[0:7]
1	1	1	0	1	V_{RL}	RSLT[0:7]
1	1	1	1	0	$(V_{RH} - V_{RL}) / 2$	RSLT[0:7]
1	1	1	1	1	TEST/RESERVED	RSLT[0:7]

Table 6–7. Multiple-Channel Conversions

S8CM	CD	CC	CB	CA	Input	Result Register
0	0	0	X	X	AN0	RSLT0
					AN1	RSLT1
					AN2	RSLT2
					AN3	RSLT3
0	0	1	X	X	AN4	RSLT0
					AN5	RSLT1
					AN6	RSLT2
					AN7	RSLT3
0	1	0	X	X	RESERVED	RSLT0
					RESERVED	RSLT1
					RESERVED	RSLT2
					RESERVED	RSLT3
0	1	1	X	X	V_{RH}	RSLT0
					V_{RL}	RSLT1
					$(V_{RH} - V_{RL}) / 2$	RSLT2
					TEST/RESERVED	RSLT3
1	0	X	X	X	AN0	RSLT0
					AN1	RSLT1
					AN2	RSLT2
					AN3	RSLT3
					AN4	RSLT4
					AN5	RSLT5
					AN6	RSLT6
					AN7	RSLT7
1	1	X	X	X	RESERVED	RSLT0
					RESERVED	RSLT1
					RESERVED	RSLT2
					RESERVED	RSLT3
					V_{RH}	RSLT4
					V_{RL}	RSLT5
					$(V_{RH} - V_{RL}) / 2$	RSLT6
					TEST/RESERVED	RSLT7

6.7.6 Conversion Timing

Total conversion time is made up of initial sample time, transfer time, final sample time, and resolution time. Initial sample time is the time during which a selected input channel is connected to the sample buffer amplifier via a sample capacitor. During transfer time, the sample capacitor is disconnected from the multiplexer, and the RC DAC array is driven by the sample buffer amp. During final sampling time, the sample capacitor and amplifier are bypassed, and the multiplexer input charges the RC DAC array directly. During resolution time, the voltage in the RC DAC array is converted to a digital value, and the value is stored in the SAR.

Initial sample time and transfer time are fixed at 2 ADC clock cycles each. Final sample time can be 2, 4, 8, or 16 ADC clock cycles, depending on the value of the STS field in ADCTL0. Resolution time is 10 cycles for 8-bit conversion and 12 cycles for 10-bit conversion.

Transfer and resolution require a minimum of 16 ADC clocks (8 µs with a 2.1-MHz ADC clock) for 8-bit resolution or 18 ADC clocks (9 µs with a 2.1-MHz ADC clock) for 10-bit resolution. If maximum final sample time (16 ADC clocks) is used, total conversion time is 15 µs for an 8-bit conversion or 16 µs for a 10-bit conversion (with a 2.1-MHz ADC clock).

Figures 6–2 and 6–3 illustrate the timing for 8- and 10-bit conversions, respectively. These diagrams assume a final sampling period of two ADC clocks.

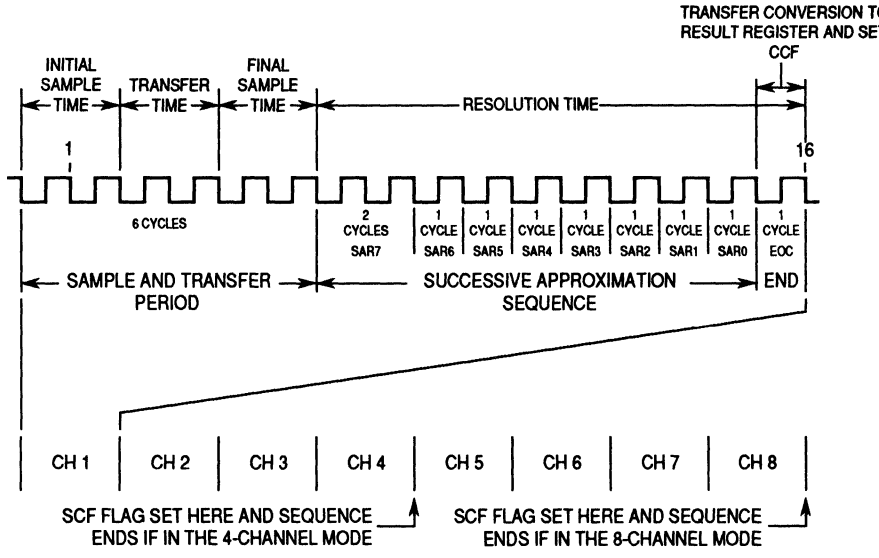


Figure 6–2. 8-Bit Conversion Timing

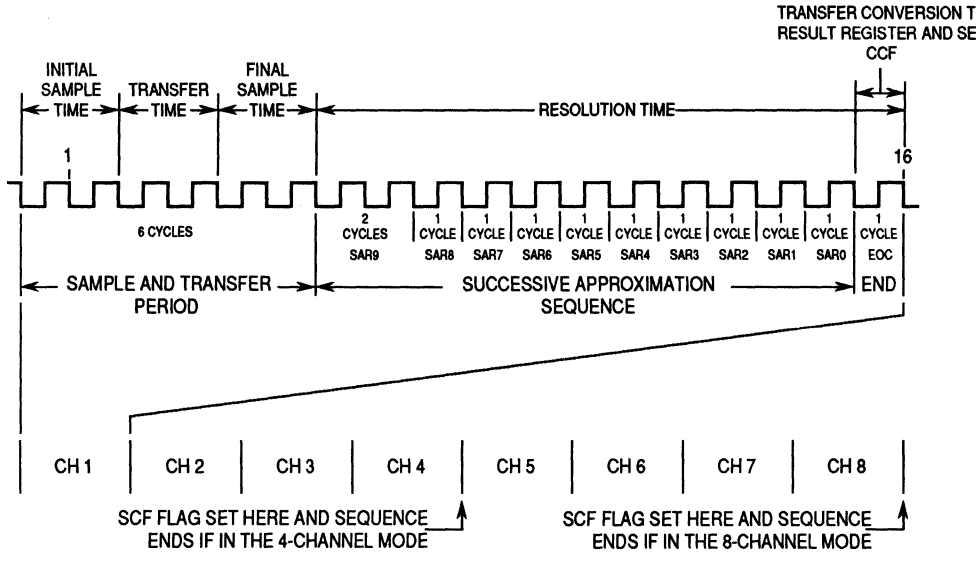


Figure 6–3. 10-Bit Conversion Timing

6.7.7 Successive Approximation Register

The successive approximation register accumulates the result of each conversion one bit at a time, starting with the most significant bit.

At the start of the resolution period, the MSB of the SAR is set, and all less significant bits are cleared. Depending on the result of the first comparison, the MSB is either left set or cleared. Each successive bit is set or left cleared in descending order until all eight or ten bits have been resolved.

When conversion is complete, the content of the SAR is transferred to the appropriate result register. Refer to **APPENDIX D REGISTER SUMMARY** for register mapping and configuration.

6.7.8 Result Registers

Result registers are used to store data after conversion is complete. The registers can be accessed from the IMB under ABIU control. Each register can be read from three different addresses in the ADC memory map. The format of the result data depends on the address from which it is read.

Unsigned Right-Justified Format — Conversion result is unsigned right-justified data. Bits [9:0] are used for 10-bit resolution, bits [7:0] are used for 8-bit conversion (bits [9:8] are zero). Bits [15:10] always return zero when read.

Signed Left-Justified Format — Conversion result is signed left-justified data. Bits [15:6] are used for 10-bit resolution, bits [15:8] are used for 8-bit conversion (bits [7:6] are zero). Although the ADC is unipolar, it is assumed that the zero point is $(V_{RH} - V_{RL}) / 2$ when this format is used. The value read from the register is an offset twos-complement number — for positive input, bit 15 = 0, for negative input, bit 15 = 1. Bits [5:0] always return zero when read.

Unsigned Left-Justified Format — Conversion result is unsigned left-justified data. Bits [15:6] are used for 10-bit resolution, bits [15:8] are used for 8-bit conversion (bits [7:6] are zero). Bits [5:0] always return zero when read.

Refer to **APPENDIX D REGISTER SUMMARY** for register mapping and configuration.

SECTION 7
QUEUED SERIAL MODULE

This section is an overview of MC68HC16Z1 queued serial module (QSM) function. Complete information on the QSM can be found in the *Queued Serial Module Reference Manual* (QSMRM/AD).

7.1 General

The QSM contains two serial interfaces, the queued serial peripheral interface (QSPI) and the serial communication interface (SCI). Figure 7–1 is a block diagram of the QSM.

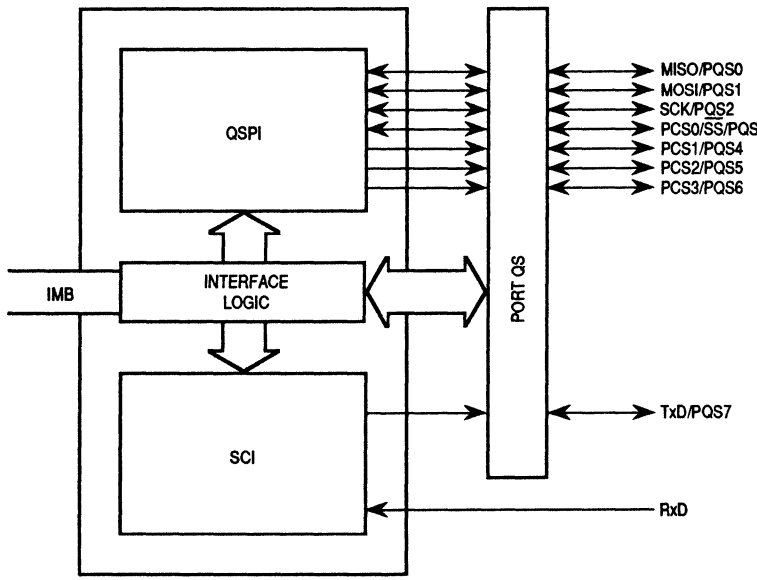


Figure 7–1. QSM Block Diagram

The QSPI provides easy peripheral expansion or interprocessor communication via 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 16 serial transfers of 8 to 16 bits each or transmission of a 256-bit data stream without CPU intervention. A special wraparound mode supports continuous sampling of a serial peripheral, with automatic QSPI RAM updating, 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). Word length of either 8 or 9 bits is software selectable. Optional parity generation and detection provide either even or odd parity check capability. Advanced error detection circuitry catches glitches of up to 1/16 of a bit time in duration. Wakeup functions allow the CPU to run uninterrupted until meaningful data is available.

7.2 QSM Registers and Address Map

There are four types of QSM registers. These are QSM global registers, QSM pin control registers, QSPI registers, and SCI registers. Global registers and pin control registers are discussed in **7.2.1 QSM Global Registers** and **7.2.2 QSM Pin Control Registers**. QSPI and SCI registers are discussed in **7.3 Queued Serial Peripheral Interface** and **7.4 Serial Communications 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 modmap (MM) bit in the system integration module configuration register (SIMCR) defines the most significant bit (ADDR23) of the IMB address for each module in the MC68HC16Z1. Because the CPU16 in the MC68HC16Z1 drives only ADDR[19:0] and ADDR[23:20] follow the logic state of ADDR19, MM must equal 1.

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

7.2.1 QSM Global Registers

The QSM configuration register (QSMCR) contains parameters for interfacing to the CPU16 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.

7.2.1.1 Low-power Stop Operation

When the STOP bit in 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 may be set by the CPU and by reset.

System software must stop the QSPI and SCI before asserting STOP in order 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 concerning low-power operation.

7.2.1.2 Freeze Operation

The freeze (FRZ[1:0]) bits in 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. See **SECTION 5 CENTRAL PROCESSING UNIT** for more information on background debugging mode.

7.2.1.3 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 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 CPU16 acknowledges an interrupt request, it places the value in the condition code register interrupt priority (IP) mask 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 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 CPU16 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.

7.2.2 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. DDRQS1 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.

The Port QS data register (PORTQS) latches I/O data. PORTQS writes drive pins defined as outputs. PORTQS reads return data present on the pins. To avoid driving undefined data, first write a byte to PORTQS, then configure DDRQS.

PQSPAR and DDRQS are 8-bit registers located at the same word address. Table 7–1 is a summary of QSM pin functions.

Table 7–1. QSM Pin Functions

	Pin	Mode	DDRQS Bit	Pin Function
QSPI Pins	MISO	Master	0	Serial Data Input to QSPI
			1	General-Purpose Digital Output
		Slave	0	General-Purpose Digital Input
			1	Serial Data Output from QSPI
	MOSI	Master	0	General-Purpose Digital Input
			1	Serial Data Output from QSPI
		Slave	0	Serial Data Input to QSPI
			1	General-Purpose Digital Output
	SCK	Master	0	General-Purpose Digital Input
			1	Clock Output from QSPI
		Slave	0	Clock Input to QSPI
			1	General-Purpose Digital Output
	PCS0/$\overline{SS}$	Master	0	Mode Fault Input
			1	Chip-Select Output
		Slave	0	QSPI Slave Select Input
			1	General-Purpose Digital Output
	PCS[3:1]	Master	0	General-Purpose Digital Input
			1	Chip-Select Output
		Slave	0	General-Purpose Digital Input
			1	General-Purpose Digital Output
SCI Pins	TXD	Transmit	X	Serial Data Output from SCI
	RXD	Receive	NA	Serial Data Input to SCI

X = DDRQS bit ignored, data is output when TE = 1

7.3 Queued Serial Peripheral Interface

The queued serial peripheral interface (QSPI) communicates with external devices via a synchronous serial bus. The QSPI is fully compatible with SPI systems found on other Motorola products, but has enhanced capabilities. The QSPI can perform full duplex three-wire or half duplex two-wire transfers. A variety of transfer rate, clocking, and interrupt-driven communication options is available.

Serial transfer of any number of bits from 8 to 16 can be specified. Programmable transfer length simplifies interfacing to a number of devices that require different data lengths.

An inter-transfer delay of 1 to 500 µs (using a 16.78-MHz system clock) may be specified (default is 1 µs). Programmable delay simplifies interfacing 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 interfacing 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 the range 8 to 256 may be transferred without CPU intervention. Longer transfers are possible, but minimal CPU intervention is required to prevent loss of data. A 1-µs pause (16.78-MHz system clock) is inserted between each queue entry transfer.

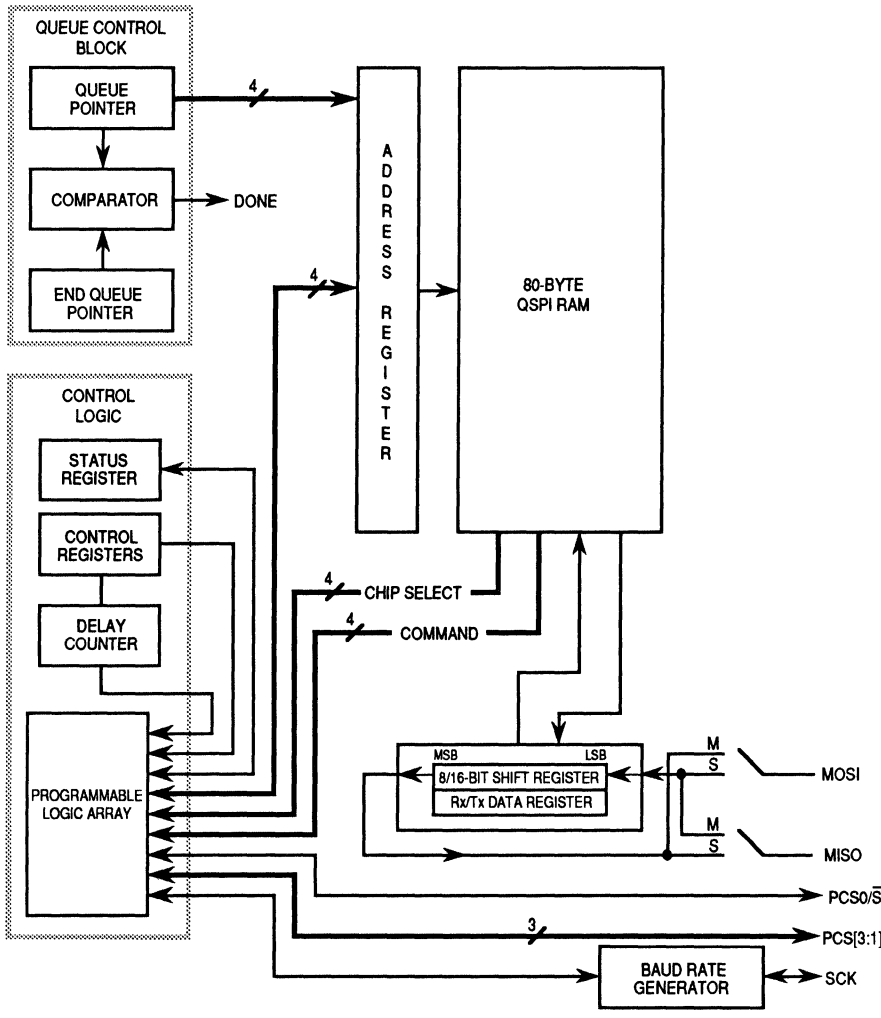


Figure 7–2. QSPI Block Diagram

7.3.1 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** register bit and field definitions.

7.3.1.1 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 access only to all bits except the SPE bit in SPCR1. Control registers must be initialized before the QSPI is enabled to insure 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.

7.3.1.2 Status Register

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

7.3.2 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. Figure 7–3 shows RAM organization.

7.3.2.1 Receive Data 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.

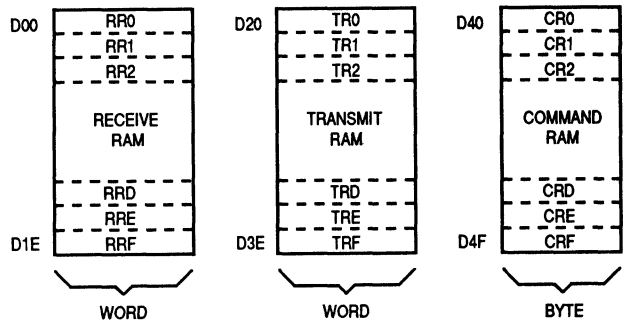


Figure 7–3. QSPI RAM

7.3.2.2 Transmit Data 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 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.

7

7.3.2.3 Command RAM

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

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

A maximum of 16 commands can be in the queue. Queue execution by the QSPI proceeds from the address in NEWQP through the address in ENDQP (both of these fields are in SPCR2).

7.3.3 QSPI Pins

The QSPI uses seven MC68HC16Z1 pins. These pins may be configured for general-purpose I/O when not needed for QSPI application.

Table 7–2 shows QSPI input and output pins and their functions.

Table 7–2. QSPI Pin Function

Pin Names	Mnemonics	Mode	Function
Master In Slave Out	MISO	Master Slave	Serial Data Input to QSPI Serial Data Output from QSPI
Master Out Slave In	MOSI	Master Slave	Serial Data Output from QSPI Serial Data Input to QSPI
Serial Clock	SCK	Master Slave	Clock Output from QSPI Clock Input to QSPI
Peripheral Chip Selects	PCS[3:0]	Master	Select Peripherals
Slave Select	$\overline{SS}$	Master Slave	Causes mode fault Initiates serial transfer

7.3.4 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 CPU16 can access three of them through fields in QSPI registers. The new queue pointer (NEWQP), contained 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), contained 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 wraparound 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.

7.3.5 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 via the QSPI. Switching between the modes is controlled by MSTR in SPCR0. Prior to entering either mode, appropriate QSM and QSPI registers must be properly initialized.

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

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

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 7–4 shows QSPI initialization; Figures 7–5 and 7–6 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 **7.6 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 may be written at any time.

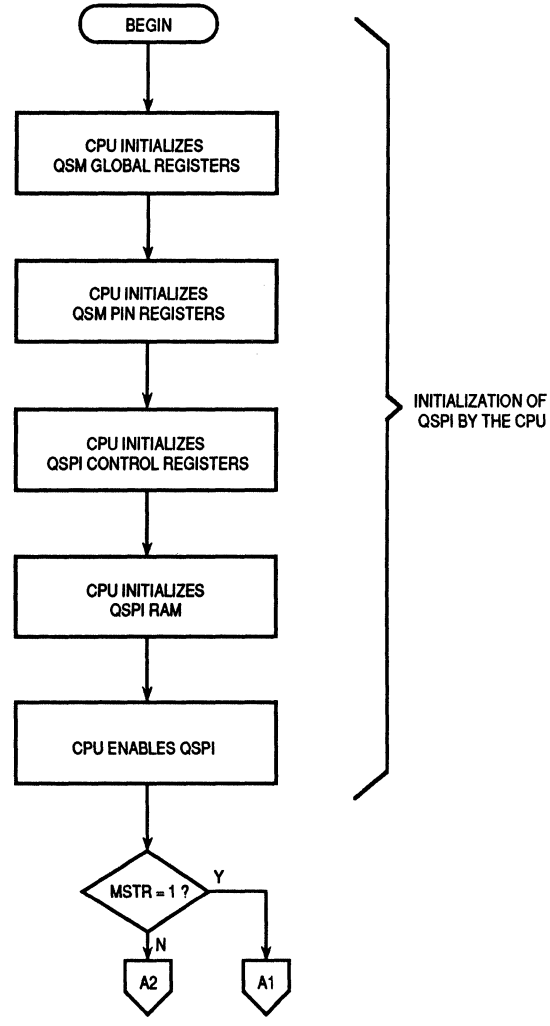


Figure 7–4. Flowchart of QSPI Initialization Operation

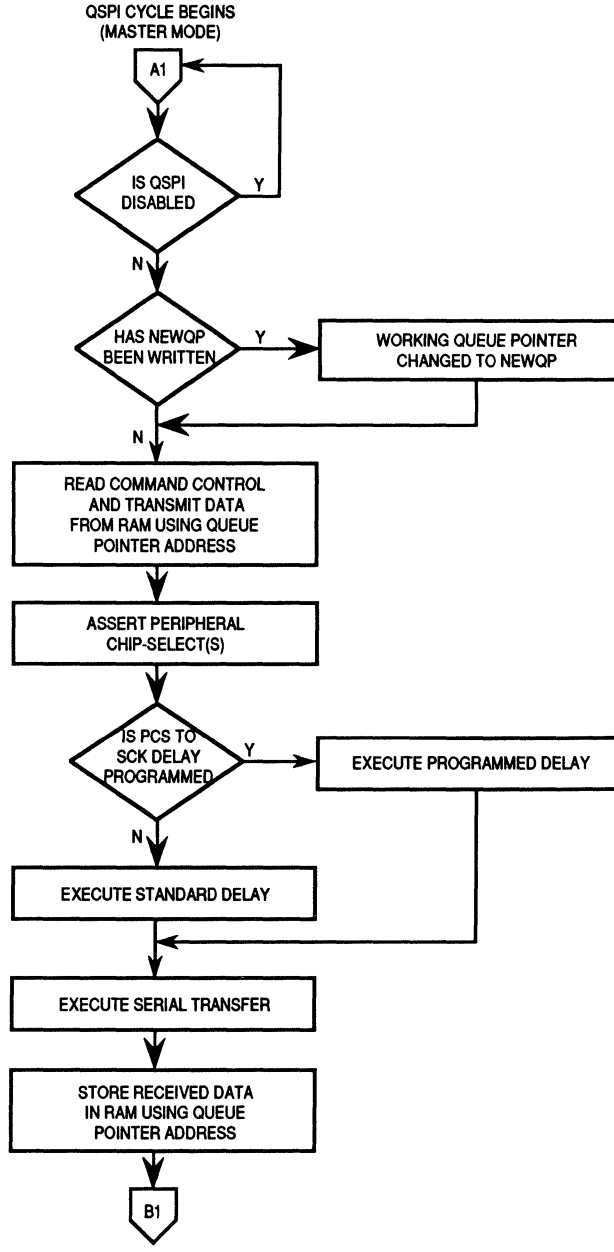


Figure 7–5. Flowchart of QSPI Master Operation (Part 1)

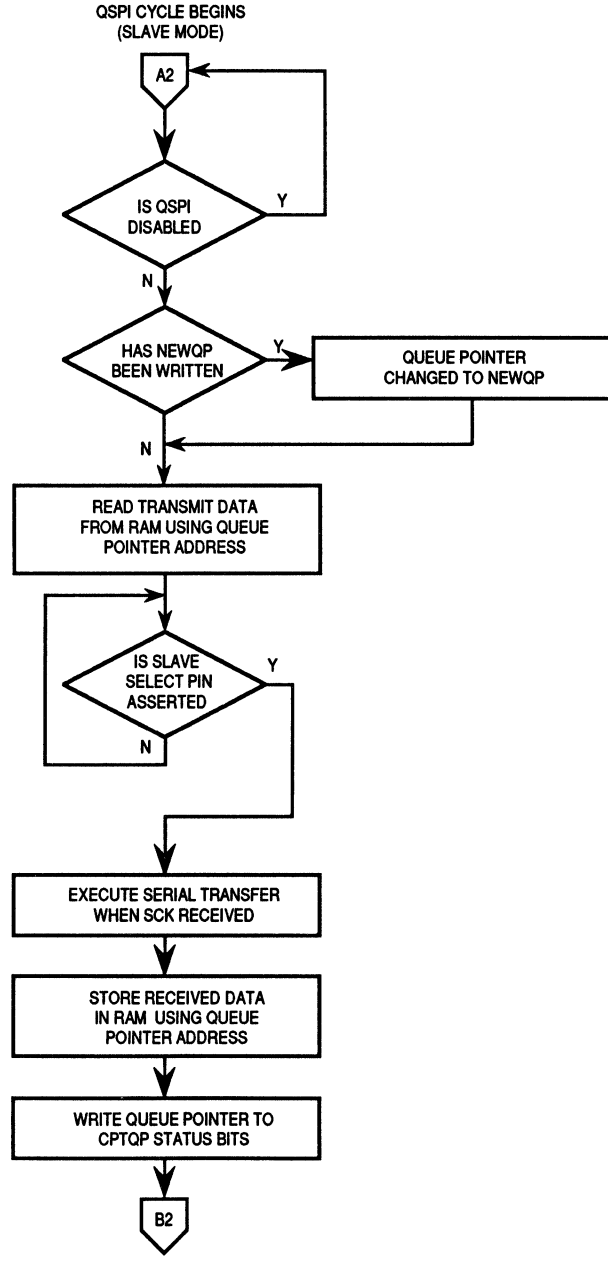


Figure 7–5. Flowchart of QSPI Master Operation (Part 2)

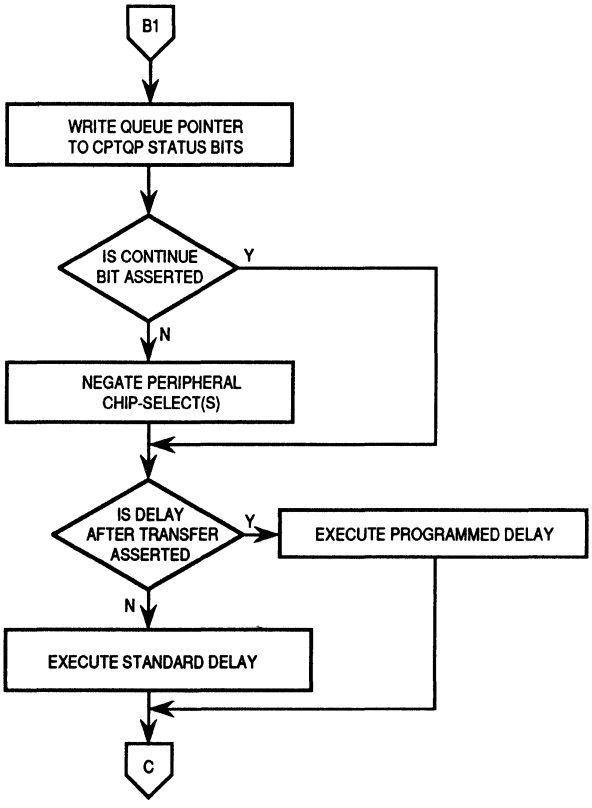


Figure 7–5. Flowchart of QSPI Master Operation (Part 3)

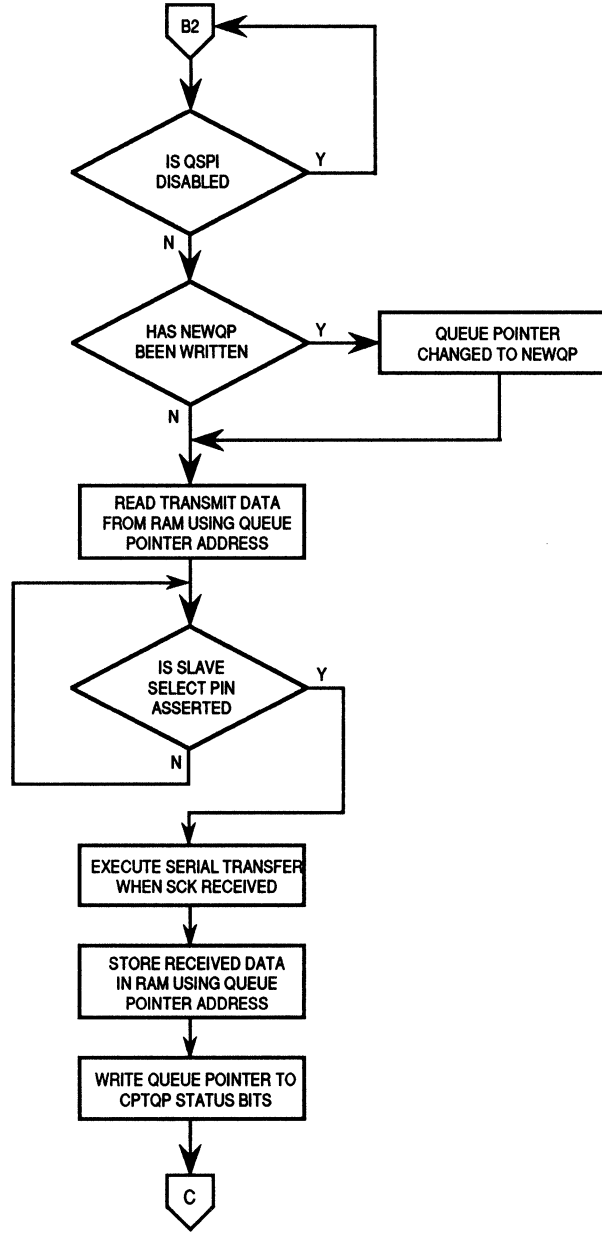


Figure 7–6. Flowchart of QSPI Slave Operation (Part 1)

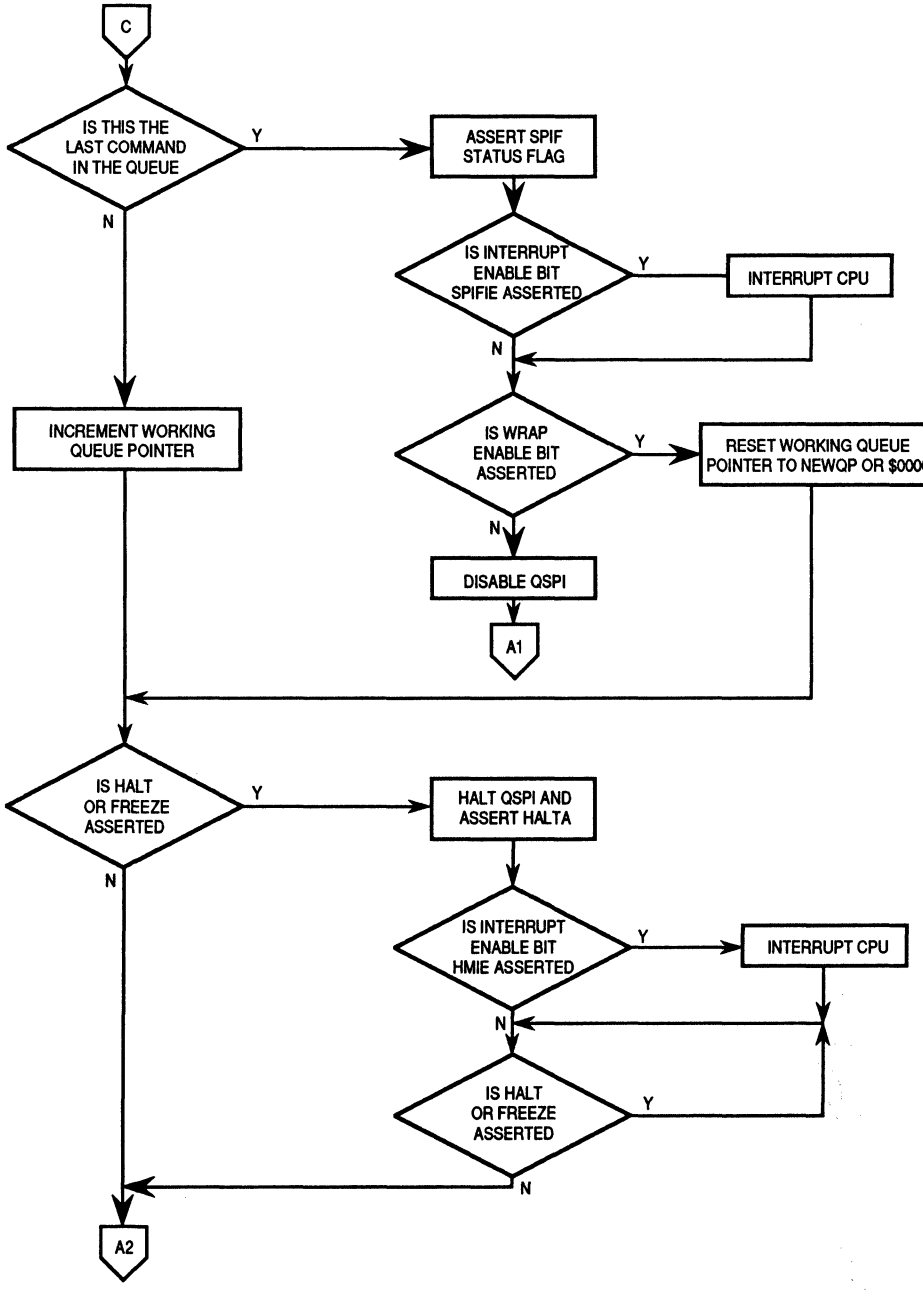


Figure 7–6. Flowchart of QSPI Slave Operation (Part 2)

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 may be specified by means of 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 may be set to any value from 8 to 16 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 pullup 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.

7.3.5.1 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:

SCK Baud Rate = System Clock/(2SPBR)

or

SPBR = System Clock/(2SCK)(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:

PCS to SCK Delay = [DSCKL/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 8 bits, or a programmed value of 8 to 16 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 7–3 shows BITS field encoding.

Table 7–3.
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 (1 µs with a 16.78-MHz system clock) 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:

Delay after Transfer = [(32DTL)/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.

Standard Delay after Transfer = [17/System Clock]

Adequate delay between transfers must be specified for long data streams. The QSPI requires time, approximately 1 µs at 16.78-MHz system clock, to load a transmit RAM entry for transfer. Receiving devices need at least 1 µs of 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 will be 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 will be 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 QPDR 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.

7.3.5.2 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 will not abort a current request.

There are two recommended methods of exiting wraparound mode. The WREN bit can be cleared, or the HALT bit in SPCR3 can be set. Exiting wraparound mode by clearing SPE is not recommended — 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.

7.3.5.3 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 commences 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 upon 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. $\overline{SS}$ need not go high between transfers — the QSPI will transfer 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.

7.3.5.4 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.

7.3.6 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.

7.4 Serial Communication Interface

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

7.4.1 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.

7.4.1.1 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.

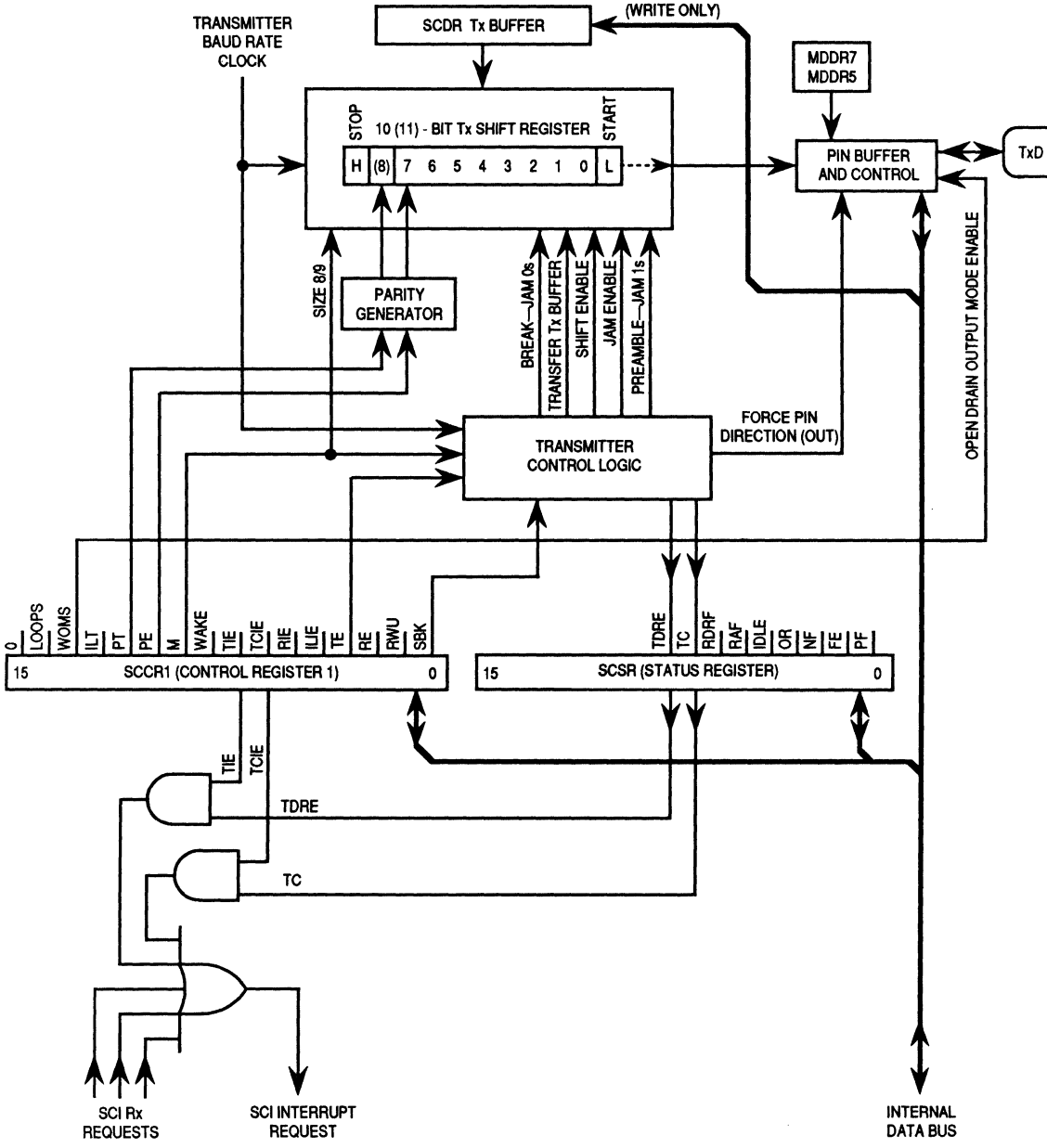


Figure 7–7. SCI Transmitter Block Diagram

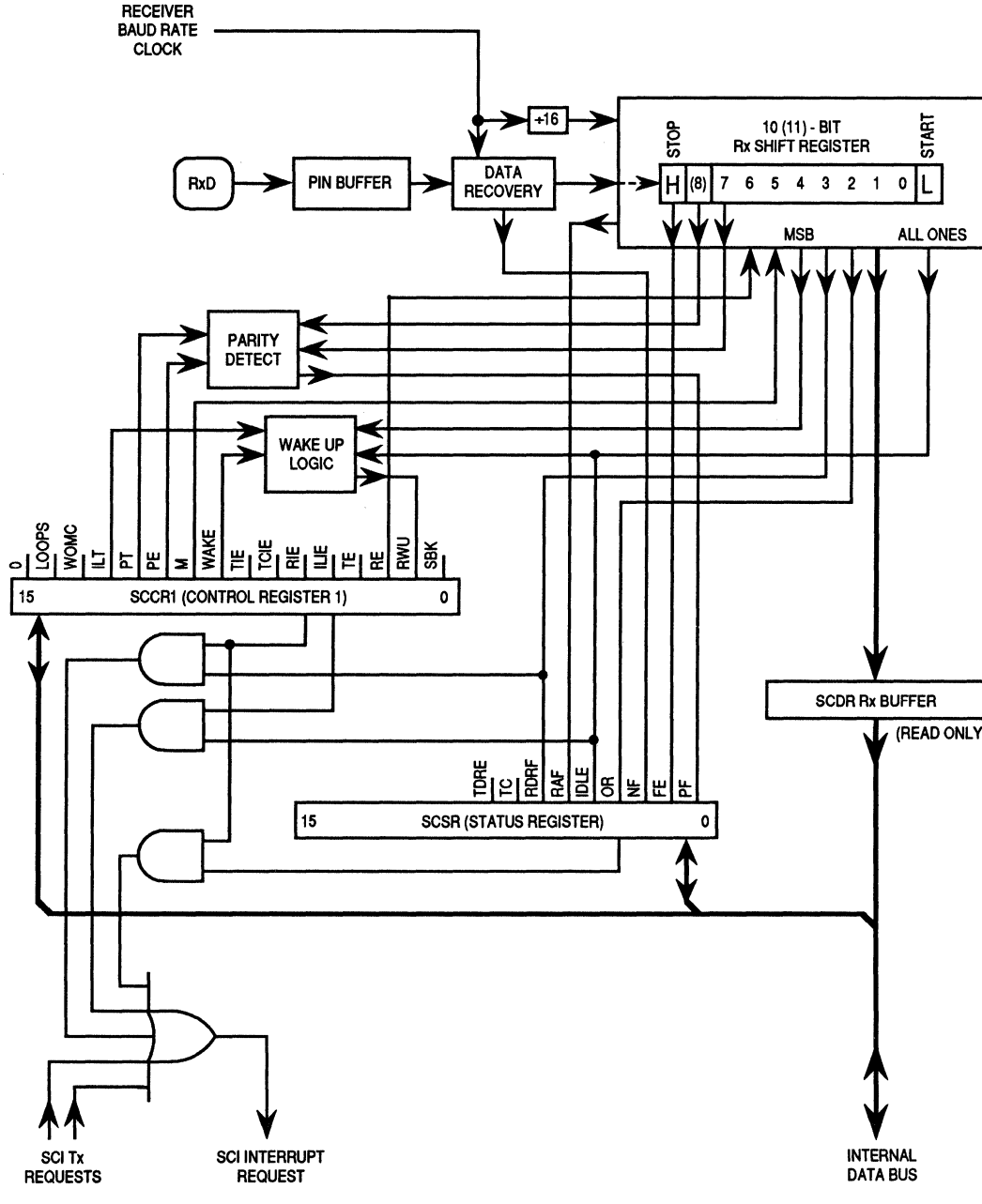


Figure 7–8. SCI Receiver Block Diagram

7.4.1.2 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 SCSR, then reading (receiver status bits) or writing (transmitter status bits) SCDR. A long-word read can consecutively access both SCSR and SCDR. This action clears receive status flag bits that were set at the time of the read, but does not clear TDRE or TC flags.

If an internal SCI signal for setting a status bit comes after the CPU has read the asserted status bits, but before the CPU has written or read register SCDR, the newly set status bit is not cleared — SCSR must be read again with the bit set, and SCDR must be written or read before the status bit is cleared.

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

7.4.1.3 Data Register

SCDR contains two data registers at the same address. 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 RDR. TDR is a write-only register that contains data to be transmitted. The 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 9-bit operation. When it is configured for 8-bit operation, they have no meaning or effect.

7.4.2 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 7–4 shows SCI pin function.

Table 7–4. 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-Purpose I/O Serial Data Output from SCI

7.4.3 SCI Operation

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

7.4.3.1 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.

7.4.3.2 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 10-bit and 11-bit frames. The serial mode (M) bit in SCI control register one (SCCR1) specifies the number of bits per frame.

The most common data frame format for NRZ serial interface is one start bit, eight data bits (LSB first), and one stop bit; a total of 10 bits. The most common 11-bit data frame contains one start bit, eight data bits, a parity or control bit, and one stop bit. 10-bit and 11-bit frames are shown in Table 7–5.

Table 7–5. 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

7.4.3.3 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:

$$\text{SCI Baud Clock Rate} = \text{System Clock}/(32\text{SCBR})$$

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.

7.4.3.4 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 7–6 shows possible data and parity formats.

Table 7–6.
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

7.4.3.5 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 — 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 will overwrite 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 PORTQS7. 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.

7.4.3.6 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 — data can 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 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 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. Since 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 CPU16 reads SCSR and SCDR in sequence, it acquires status and data, and also clears the status flags. Reading SCSR acquires status and arms the clearing mechanism. Reading SCDR acquires data and clears 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.

7.4.3.7 Idle-Line Detection

During a typical serial transmission, frames are transmitted isochronously — 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 will cause 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.

7.4.3.8 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 wake up (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 due to an additional bit-time per frame.

7.4.3.9 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 prior to entering loop mode.

7.5 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, 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).

7

SECTION 8
GENERAL-PURPOSE TIMER

The general-purpose timer (GPT) is an 11-channel timer for use in systems where a moderate level of CPU control is required. This section is an overview of GPT function. Refer to the *GPT Reference Manual* (GPTRM/AD) for complete information on the GPT module.

8.1 General

The GPT consists of a capture/compare unit, a pulse accumulator, and two pulse-width modulators. A bus interface unit connects the GPT to the intermodule bus (IMB).

The capture/compare unit features three input capture channels, four output compare channels, and one channel that can be selected as an input capture or output compare channel. These channels share a 16-bit free-running counter (TCNT) which derives its clock from a nine-stage prescaler or from the external clock input signal, PCLK.

Pulse accumulator channel logic includes an 8-bit counter; the pulse accumulator can operate in either event counting mode or gated time accumulation mode.

Pulse-width modulator outputs are periodic waveforms whose duty cycles can be independently selected and modified by user software. The PWM circuits share a 16-bit free-running counter that can be clocked by the same nine-stage prescaler used by the capture/compare unit or by the PCLK input.

All GPT pins can also be used for general-purpose input/output. The input capture and output compare pins form a bidirectional 8-bit parallel port (Port GP). PWM pins are outputs only. PAI and PCLK pins are inputs only.

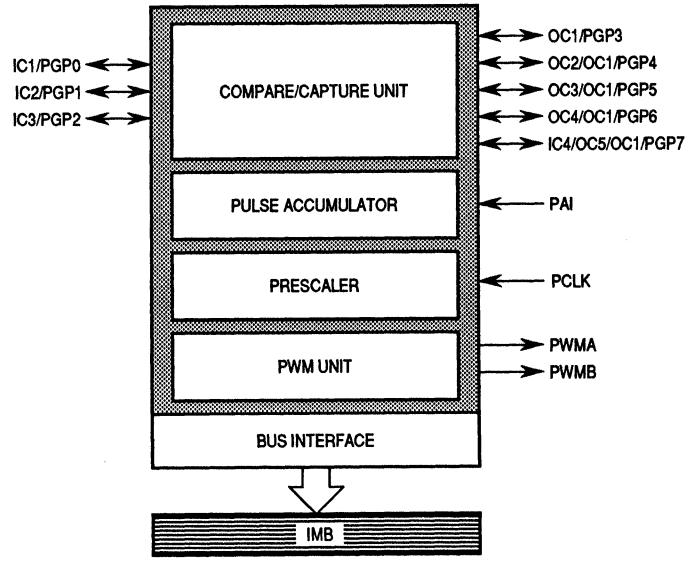


Figure 8–1. GPT Block Diagram

8

8.2 GPT Registers and Address Map

The GPT programming model consists of a configuration register (GPTMCR), parallel I/O registers (DDRGP, PORTGP), capture/compare registers (TCNT, TCTL1, TCTL2, TIC[1:3], TOC[1:4], TI4/O5, CFORC), pulse accumulator registers (PACNT, PACTL), pulse-width modulation registers (PWMA, PWMB, PWMC, PWMCNT, PWMBUFA, PWMBUFB), status registers (TFLG1, TFLG2) and interrupt control registers (TMSK1, TMSK2). Functions of the module configuration register are discussed in **8.3 Special Modes of Operation** and **8.4 Polled and Interrupt-Driven Operation**. Other register functions are discussed in the appropriate sections.

All registers can be accessed using byte or word operations. Certain capture/compare registers and pulse-width modulation registers must be accessed by word operations to ensure coherency. If byte accesses are used to read a register such as the timer counter register (TCNT), there is a possibility that data in the byte not being accessed will change while the other byte is read. Both bytes must be accessed at the same time.

The modmap (MM) bit in the system integration module configuration register (SIMCR) defines the most significant bit (ADDR23) of the IMB address for each register in the MCU. Because the CPU16 drives only ADDR[19:0] and ADDR[23:20] follow the logic state of ADDR19, MM must equal one.

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

8.3 Special Modes of Operation

The GPT module configuration register (GPTMCR) is used to control special GPT operating modes. These include low-power stop mode, freeze mode, single-step mode, and test mode. Normal GPT operation can be polled or interrupt-driven. Refer to **8.4 Polled and Interrupt-Driven Operation** for more information.

8.3.1 Low-power Stop Mode

Low-power stop operation is initiated by setting the STOP bit in GPTMCR. In stop mode the system clock to the module is turned off. The clock remains off until STOP is negated or a reset occurs. All counters and prescalers within the timer stop counting while the STOP bit is set. Only the module configuration register (GPTMCR) and the interrupt configuration register (ICR) should be accessed while in the stop mode. Accesses to other GPT registers cause unpredictable behavior. Low-power stop can also be used to disable module operation during debugging.

8.3.2 Freeze Mode

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

Freeze mode freezes the current state of the timer. The prescaler and the pulse accumulator do not increment and changes to the pins are ignored (input pin synchronizers are not clocked). All of the other timer functions that are controlled by the CPU will operate normally; for example, registers can be written to change pin directions, force output compares, and read or write I/O pins.

While the FREEZE signal is asserted, the CPU has write access to registers and bits that are normally read-only, or write-once. The write-once bits can be written to as often as needed. The prescaler and the pulse accumulator remain stopped and the input pins are ignored until the FREEZE signal is negated (the CPU is no longer in BDM), the FRZ0 bit is cleared, or the MCU is reset.

Activities that are in progress prior to FREEZE assertion are completed. For example, if an input edge on an input capture pin is detected just as the FREEZE signal is asserted, the capture occurs and the corresponding interrupt flag is set.

8.3.3 Single-Step Mode

Two bits in GPTMCR support GPT debugging without using BDM. When the STOPP bit is asserted, the prescaler and the pulse accumulator stop counting and changes at input pins are ignored. Reads of the GPT pins will return the state of the pin when STOPP was set. After STOPP is set, the INCP bit can be set to increment the prescaler and clock the input synchronizers once. The INCP bit is self-negating after the prescaler is incremented. INCP can be set repeatedly. The INCP bit has no effect when the STOPP bit is not set.

8.3.4 Test Mode

Test mode is used during Motorola factory testing. The GPT has no dedicated test-mode control register; all GPT testing in the MC68HC16Z1 is done under control of the system integration module.

8

8.4 Polled and Interrupt-Driven Operation

Normal GPT function can be polled or interrupt-driven. All GPT functions have an associated status flag and an associated interrupt. The timer interrupt flag registers (TFLG1 and TFLG2) contain status flags used for polled and interrupt-driven operation. The timer mask registers (TMSK1 and TMSK2) contain interrupt control bits. Control routines can monitor GPT operation by polling the status registers. When an event occurs, the control routine transfers control to a service routine that handles that event. If interrupts are enabled for an event, the GPT requests interrupt service when the event occurs. Using interrupts does not require continuously polling the status flags to see if an event has taken place. However, status flags must be cleared after an interrupt is serviced, in order to disable the interrupt request.

8.4.1 Polled Operation

When an event occurs in the GPT, that event sets a status flag in TFLG1 or TFLG2. The GPT sets the flags; they cannot be set by the CPU. TFLG1 and TFLG2 are 8-bit registers that can be accessed individually or as one 16-bit register. The registers are initialized to zero at reset. Table 8–1 shows status flag assignment.

Table 8–1. GPT Status Flags

Flag Mnemonic	Register Assignment	Source
IC1F	TFLG1	Input Capture 1
IC2F	TFLG1	Input Capture 2
IC3F	TFLG1	Input Capture 3
OC1F	TFLG1	Output Compare 1
OC2F	TFLG1	Output Compare 2
OC3F	TFLG1	Output Compare 3
OC4F	TFLG1	Output Compare 4
IC4/OC5F	TFLG1	Input Capture 4/Output Compare 5
TOF	TFLG2	Timer Overflow
PAOVF	TFLG2	Pulse Accumulator Overflow
PAIF	TFLG2	Pulse Accumulator Input

For each bit in TFLG1 and TFLG2 there is a corresponding bit in TMSK1 and TMSK2 in the same bit position. If a mask bit is set and an associated event occurs, a hardware interrupt request is generated.

In order to re-enable a status flag after an event occurs, the status flags must be cleared. Status registers are cleared in a particular sequence. The register must first be read for set flags, then zeros must be written to the flags that are to be cleared. If a new event occurs between the time that the register is read and the time that it is written, the associated flag is not cleared.

8.4.2 GPT Interrupts

The GPT has 11 internal sources that can cause it to request interrupt service (refer to Table 8–2). Setting bits in TMSK1 and TMSK2 enables specific interrupt sources. TMSK1 and TMSK2 are 8-bit registers that can be addressed individually or as one 16-bit register. The registers are initialized to zero at reset. For each bit in TMSK1 and TMSK2 there is a corresponding bit in TFLG1 and TFLG2 in the same bit position. TMSK2 also controls the operation of the timer prescaler. Refer to **8.7 Prescaler** for more information.

The value of the interrupt level (IRL) field in the interrupt control register (ICR) determines the priority of GPT interrupt requests. IRL values correspond to MCU interrupt request signals $\overline{IRQ}[7:1]$. $\overline{IRQ7}$ is the highest priority interrupt request signal; $\overline{IRQ1}$ is the lowest-priority signal. A value of %111 causes $\overline{IRQ7}$ to be asserted when a GPT interrupt request is made; lower field values cause corresponding lower-priority interrupt request signals to be asserted. Setting field value to %000 disables interrupts.

Table 8–2. GPT Interrupt Sources

Name	Source Number	Source	Vector Number
—	0000	Adjusted Channel	IVBA : 0000
IC1	0001	Input Capture 1	IVBA : 0001
IC2	0010	Input Capture 2	IVBA : 0010
IC3	0011	Input Capture 3	IVBA : 0011
OC1	0100	Output Compare 1	IVBA : 0100
OC2	0101	Output Compare 2	IVBA : 0101
OC3	0110	Output Compare 3	IVBA : 0110
OC4	0111	Output Compare 4	IVBA : 0111
IC4/OC5	1000	Input Capture 4/Output Compare 5	IVBA : 1000
TO	1001	Timer Overflow	IVBA : 1001
PAOV	1010	Pulse Accumulator Overflow	IVBA : 1010
PAI	1011	Pulse Accumulator Input	IVBA : 1011

The CPU16 recognizes only interrupt request signals of a priority greater than the condition code register interrupt priority (IP) mask value. When the CPU acknowledges an interrupt request, the priority of the acknowledged request is written to the IP mask and driven out on the IMB address lines.

When the IP mask value driven out on the address lines is the same as the IRL value, the GPT contends for arbitration priority. GPT arbitration priority is determined by the value of the IARB field in GPTMCR. Each MCU module that can make interrupt requests must be assigned a nonzero IARB value in order to implement an arbitration scheme. Arbitration is performed by means of serial assertion of IARB field bit values.

When the GPT 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 CPU16 exception vector table. Vector numbers are formed by concatenating the value in the ICR IVBA field with a 4-bit value supplied by the GPT when an interrupt request is made.

Hardware prevents the vector number from changing while it is being driven out on the IMB. Vector number assignment is shown in Table 8–2.

At reset, IVBA is initialized to $0. To enable interrupt-driven timer operation, the upper nibble of a user-defined vector number ($40–$FF) must be written to IVBA, and interrupt handler routines must be located at the addresses pointed to by the corresponding vector. Note that IVBA must be written before GPT interrupts are enabled, or the GPT could supply a vector number ($00 to $0F) that corresponds to an assigned or reserved exception vector.

The internal GPT interrupt priority hierarchy is shown in Table 8–2. The lower the interrupt source number, the higher the priority. A single GPT interrupt source can be given priority over all other GPT interrupt sources by assigning the priority adjust field (PAB) in the ICR a value equal to its source number.

Interrupt requests are asserted until associated status flags are cleared. Status flags must be cleared in a particular sequence. The status register must first be read for set flags, then zeros must be written to the flags that are to be cleared. If a new event occurs between the time that the register is read and the time that it is written, the associated flag is not cleared.

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

8.5 Pin Descriptions

The GPT uses twelve of the MC68HC16Z1 pins. Each pin can perform more than one function. Descriptions of GPT pins divided into functional groups follow.

8.5.1 Input Capture Pins (IC[1:3])

Each of these pins is associated with a single GPT input capture function. Each pin has hysteresis — any pulse longer than two system clocks is guaranteed to be valid and any pulse shorter than one system clock is ignored. Each pin has an associated 16-bit capture register that holds the captured counter value. These pins can also be used for general-purpose I/O. Refer to **8.8.2 Input Capture Functions** for more information.

8.5.2 Input Capture/Output Compare Pin (IC4/OC5)

This pin can be configured for use by either an input capture or an output compare function. It has an associated 16-bit register that is used for holding either the input capture value or the output match value. When used for input capture the pin has the same hysteresis as other input capture pins. The pin can be used for general-purpose I/O. Refer to **8.8.2 Input Capture Functions** and **8.8.3 Output Compare Functions** for more information.

8.5.3 Output Compare Pins (OC[1:4])

These pins are used for GPT output compare functions. Each pin has an associated 16-bit compare register and a 16-bit comparator. Pins OC2, OC3, and OC4 are associated with a specific output compare function. The OC1 function can affect the output of all compare pins. If the OC1 pin is not needed for an output compare function it can be used to output the clock selected for the timer counter register. Any of these pins can also be used for general-purpose I/O. Refer to **8.8.3 Output Compare Functions** for more information.

8.5.4 Pulse Accumulator Input Pin (PAI)

The PAI pin connects a discrete signal to the pulse accumulator for timed or gated pulse accumulation. PAI has hysteresis — any pulse longer than two system clocks is guaranteed to be valid and any pulse shorter than one system clock is ignored. It can be used as a general-purpose input pin. Refer to **8.10 Pulse Accumulator** for more information.

8.5.5 Pulse-Width Modulation (PWMA, PWMB)

PWMA and PWMB pins carry pulse-width modulator outputs. The modulators can be programmed to generate a periodic waveform of variable frequency and duty cycle. PWMA can be used to output the clock selected as the input to the PWM counter. These pins can also be used for general-purpose output. Refer to **8.11 Pulse-Width Modulation (PWM) Unit** for more information.

8.5.6 Auxiliary Timer Clock Input (PCLK)

PCLK connects an external clock to the GPT. The external clock can be used as the clock source for the capture/compare unit or the PWM unit in place of one of the prescaler outputs. PCLK has hysteresis — any pulse longer than two system clocks is guaranteed to be valid and any pulse shorter than one system clock is ignored. This pin can also be used as a general-purpose input pin. Refer to **8.7 Prescaler** for more information.

8.6 General-Purpose I/O

Any GPT pin can be used for general-purpose I/O when it is not used for another purpose. Capture/compare pins are bidirectional, others can be used only for output or input. I/O direction is controlled by a data direction bit in the Port GP data direction register (DDRGP).

Parallel data is read from and written to the Port GP data register (PORTGP). Pin data can be read even when pins are configured for a timer function. Data read from PORTGP always reflects the state of the external pin, while data written to PORTGP may not always affect the external pin.

Data written to PORTGP does not immediately affect pins used for output compare functions, but the data is latched. When an output compare function is disabled, the last data written to PORTGP is driven out on the associated pin if it is configured as an output. Data written to PORTGP can cause input captures if the corresponding pin is configured for input capture function.

The pulse accumulator input (PAI) and the external clock input (PCLK) pins provide general-purpose input. The state of these pins can be read by accessing the PAIS and PCLKS bits in the pulse accumulator control register (PACTL).

Pulse-width modulation A and B (PWMA/PWMB) output pins can serve as general-purpose outputs. The force PWM value (FPWMx) and the force logic one (F1x) bits in the compare force (CFORC) and PWM control (PWMC) registers, respectively, control their operation.

8.7 Prescaler

Capture/compare and PWM units have independent 16-bit free-running counters as a main timing component. These counters derive their clocks from the prescaler or from the PCLK input. Figure 8–2 is a prescaler block diagram.

In the prescaler, the system clock is divided by a nine-stage divider chain. Prescaler outputs equal to system clock divided by 2, 4, 8, 16, 32, 64, 128, 256 and 512 are provided. Connected to these outputs are 2 multiplexers, one for the capture/compare unit, the other for the PWM unit.

Multiplexers can each select one of seven prescaler taps or an external input from the PCLK pin. Multiplexer output for the timer counter (TCNT) is selected by bits CPR[2:0] in timer interrupt mask register 2 (TMSK2). Multiplexer output for the PWM counter (PWMCNT) is selected by bits PPR[2:0] in the PWM control register C (PWMC).

After reset, the GPT is configured to use system clock divided by 4 for TCNT and system clock divided by 2 for PWMCNT. Initialization software can change the division factor — the PPR bits can be written at any time but the CPR bits can only be written once after reset unless the GPT is in test or freeze mode.

The prescaler can be read at any time. In freeze mode the prescaler can also be written. Word accesses must be used to ensure coherency. If coherency is not needed byte accesses may be used. The prescaler value is contained in bits [8:0] while bits [15:9] are unimplemented and will be read as zeros.

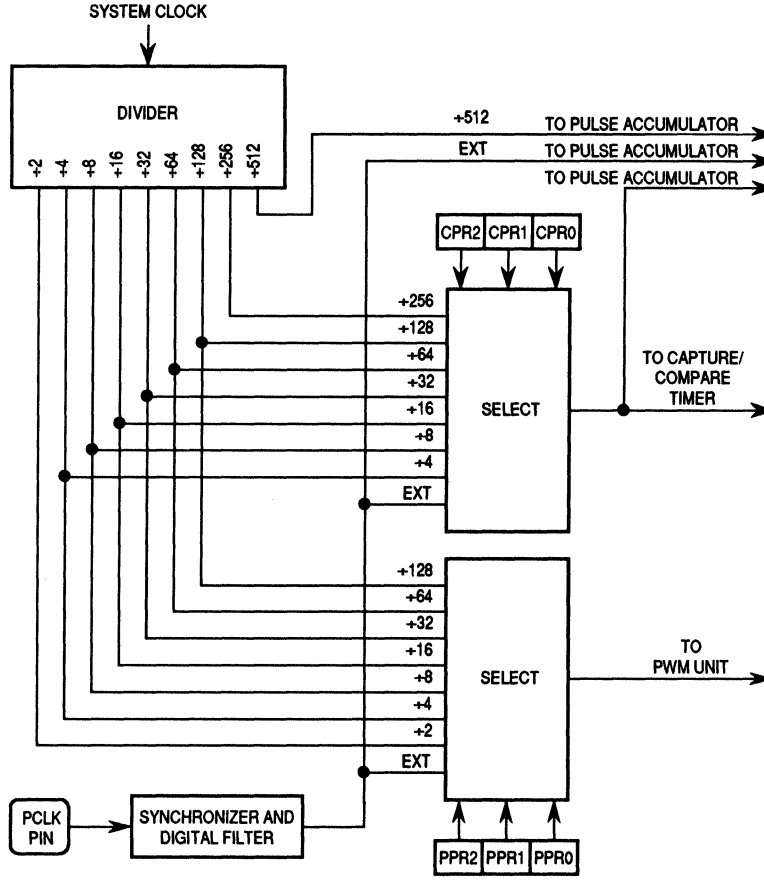


Figure 8–2. Prescaler Block Diagram

Multiplexer outputs (including the PCLK signal) can be connected to external pins. The CPROUT bit in the TMSK2 register configures the OC1 pin to output the TCNT clock and the PPROUT bit in the PWMC register configures the PWMA pin to output the PWMC clock. CPROUT and PPROUT can be written at any time. Clock signals on OC1 and PWMA do not have a 50% duty cycle. They have the period of the selected clock but are high for only one system clock time.

The prescaler also supplies three clock signals to the pulse accumulator clock select mux. These are the system clock divided by 512, the external clock signal from the PCLK pin and the capture/compare clock signal.

8.8 Capture/Compare Unit

The capture/compare unit contains the timer counter (TCNT), the input capture (IC) functions and the output compare (OC) functions. Figure 8–3 is a block diagram of the capture/compare unit.

8.8.1 Timer Counter

The timer counter (TCNT) is the key timing component in the capture/compare unit. The timer counter is a 16-bit free-running counter that starts counting after the processor comes out of reset. The counter cannot be stopped during normal operation. After reset, the GPT is configured to use the system clock divided by four as the input to the counter. The prescaler divides the system clock and provides selectable input frequencies. User software can configure the system to use one of seven prescaler outputs or an external clock.

The counter can be read any time without affecting its value. Because the GPT is interfaced to the IMB and the IMB supports a 16-bit bus, a word read gives a coherent value. If coherency is not needed, byte accesses can be made. The counter is set to $0000 during reset and is normally a read-only register. In test mode and freeze mode, any value can be written to the timer counter.

When the counter rolls over from $FFFF to $0000, the timer overflow flag (TOF) in timer interrupt flag register 2 (TFLG2) is set. An interrupt can be enabled by setting the corresponding interrupt enable bit (TOI) in timer interrupt mask register 2 (TMSK2). Refer to **8.4.2 GPT Interrupts** for more information.

8.8.2 Input Capture Functions

All GPT input capture functions use the same 16-bit timer counter (TCNT). Each input capture pin has a dedicated 16-bit latch and input edge-detection/selection logic. Each input capture function has an associated status flag, and can cause the GPT to make an interrupt service request.

When a selected edge transition occurs on an input capture pin, the associated 16-bit latch captures the content of TCNT and sets the appropriate status flag. An interrupt request can be generated when the transition is detected.

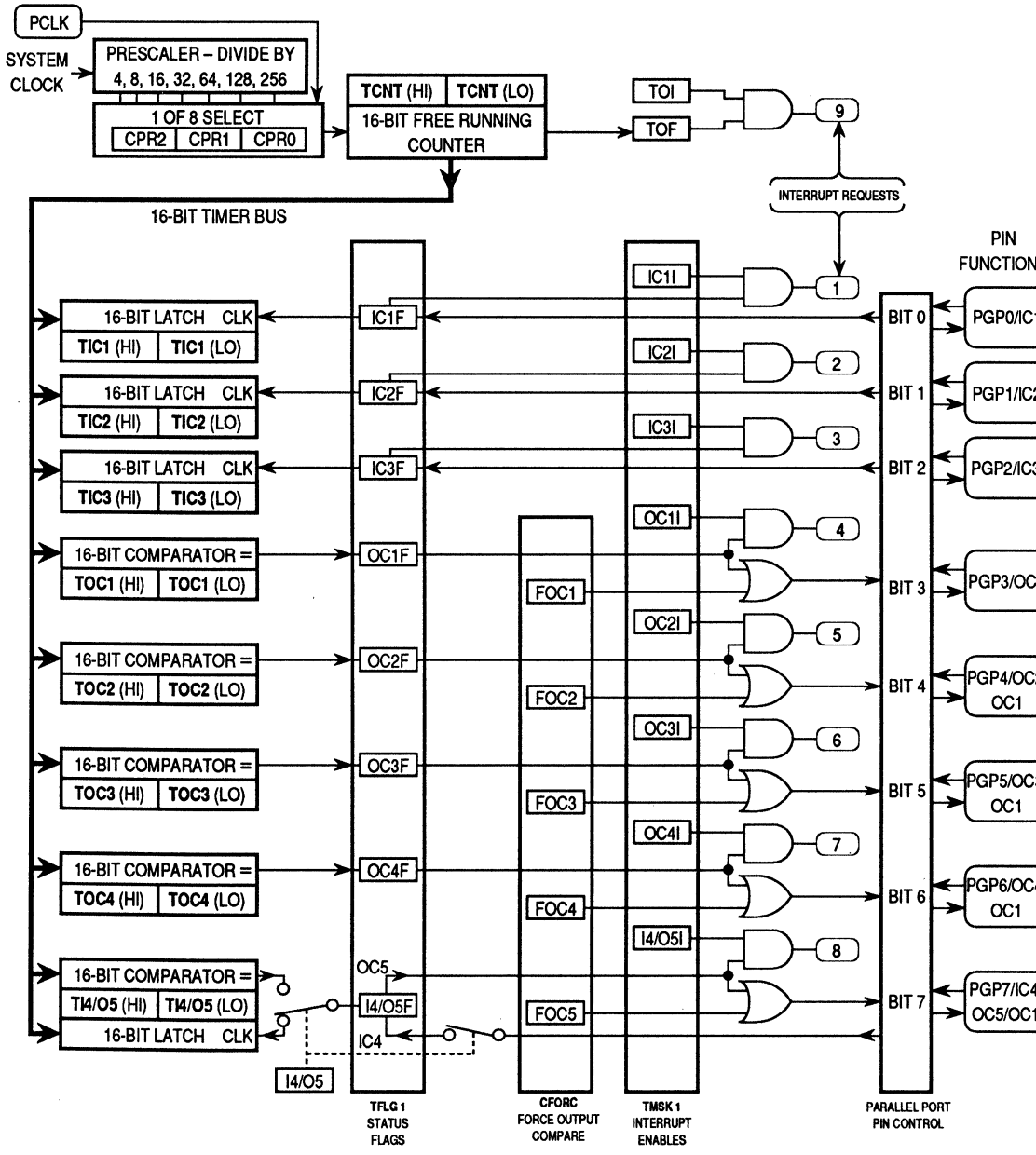


Figure 8–3. Capture/Compare Block Unit Diagram

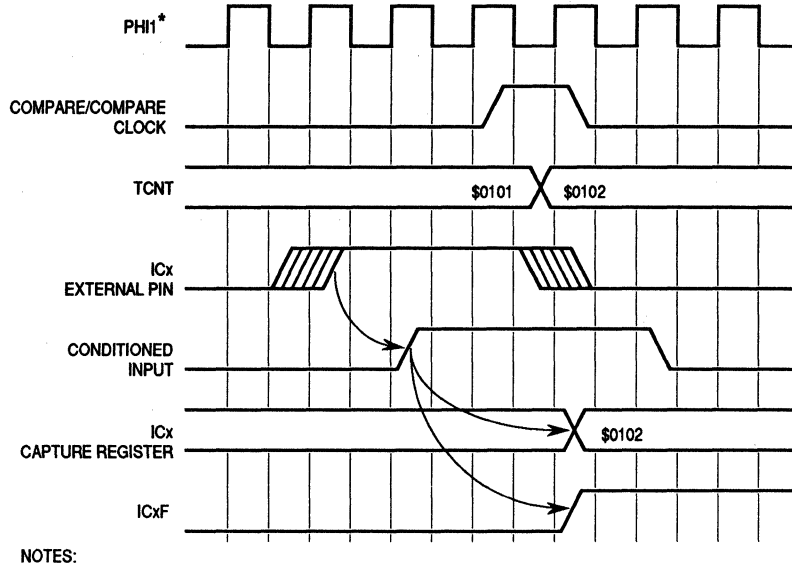
Edge-detection logic consists of control bits that enable edge detection and select a transition to detect. The EDGExA and EDGExB bits in timer control register 2 (TCTL2) determine whether the input capture functions detect rising edges only, falling edges only, or both rising and falling edges. Clearing both bits disables the input capture function. Input capture functions operate independently of each other and can capture the same TCNT value if individual input edges are detected within the same timer count cycle.

Input capture interrupt logic includes a status flag, which indicates that an edge has been detected, and an interrupt enable bit. An input capture event sets the ICxF bit in the timer interrupt flag register 1 (TFLG1) and causes the GPT to make an interrupt request if the corresponding ICxI bit is set in the timer interrupt mask register 1 (TMSK1). If the ICxI bit cleared, software must poll the status flag to determine that an event has occurred. Refer to **8.4 Polled and Interrupt-Driven Operation** for more information.

Input capture events are generally asynchronous to the timer counter. Because of this, input capture signals are conditioned by a synchronizer and digital filter. Events are synchronized with the system clock so that latching of TCNT content and counter incrementation occur on opposite half-cycles of the system clock. Inputs have hysteresis — capture of any transition longer than two system clocks is guaranteed; any transition shorter than one system clock has no effect.

Figure 8–4 shows the relationship of system clock to synchronizer output. The value latched into the capture register is the value of the counter several system clock cycles after the transition that triggers the edge detection logic. There can be up to one clock cycle of uncertainty in latching of the input transition. Maximum time is determined by the system clock frequency.

The input capture register is a 16-bit register — a word access is required to ensure coherency. If coherency is not required, byte accesses can be used to read the register. Input capture registers can be read at any time without affecting their values.



NOTES:
The conditioned input signal causes the current value of TCNT to be latched by the ICx capture register. The ICxF flag is set at the same time.
*PHI1 is the same frequency as the system clock; however, it does not have the same timing.

Figure 8–4. Input Capture Timing Example

An input capture occurs every time a selected edge is detected, even when the input capture status flag is set. This means that the value read from the input capture register corresponds to the most recent edge detected, which may not be the edge that caused the status flag to be set.

8

8.8.3 Output Compare Functions

Each GPT output compare pin has an associated 16-bit compare register and a 16-bit comparator. Each output compare function has an associated status flag, and can cause the GPT to make an interrupt service request. Output compare logic is designed to prevent false compares during data transition times.

When the programmed content of an output compare register matches the value in TCNT, an output compare status flag (OCxF) bit in TFLG1 is set. If the appropriate interrupt enable bit (OCxI) in TMSK1 is set, an interrupt request is made when a match occurs. Refer to **8.4.2 GPT Interrupts** for more information.

Operation of output compare 1 differs from that of the other output compare functions. OC1 control logic can be programmed to make state changes on other OC pins when an OC1 match occurs. Control bits in the timer compare force register (CFORC) allow for early forced compares.

8.8.3.1 Output Compare 1

Output compare 1 can affect any or all of OC[1:5] when an output match occurs. In addition to allowing generation of multiple control signals from a single comparison operation, this function makes it possible for two or more output compare functions to control the state of a single OC pin. Output pulses as short as one timer count can be generated in this way.

The OC1 action mask register (OC1M) and the OC1 action data register (OC1D) control OC1 function. Setting a bit in OC1M selects a corresponding bit in the GPT parallel data port. Bits in OC1D determine whether selected bits are to be set or cleared when an OC1 match occurs. Pins must be configured as outputs in order for the data in the register to be driven out on the corresponding pin. If an OC1 match and another output match occur at the same time and both attempt to alter the same pin, the OC1 function controls the state of the pin.

8.8.3.2 Forced Output Compare

Timer compare force register (CFORC) is used to make forced compares. The action taken as a result of a forced compare is the same as when an output compare match occurs, except that status flags are not set. Forced channels take programmed actions immediately after the write to CFORC.

The CFORC register is implemented as the upper byte of a 16-bit register which also contains the PWM control register C (PWMC). It can be accessed as 8 bits or a word access can be used. Reads of force compare bits (FOC) have no meaning and always return zeros. These bits are self-negating.

8.9 Input Capture 4/Output Compare 5

The IC4/OC5 pin can be used for input capture, output compare, or general-purpose I/O. A function enable bit (I4/O5) in the pulse accumulator control register (PACTL) configures the pin for input capture (IC4) or output compare function (OC5). Both bits are cleared during reset, configuring the pin as an input, but also enabling the OC5 function. IC4/OC5 I/O functions are controlled by the DDRGPI4/O5 bit in the Port GP data direction register (DDRGP).

The 16-bit register (TI4/O5) used with the IC4/OC5 function acts as an input capture register or as an output compare register depending on which function is selected. When used as the input capture 4 register, it cannot be written to except in test or freeze mode.

8.10 Pulse Accumulator

The pulse accumulator counter (PACNT) is an 8-bit read/write up-counter. PACNT can operate in external event counting or gated time accumulation modes. Figure 8–5 is a block diagram of the pulse accumulator.

In event counting mode, the counter increments each time a selected transition of the pulse accumulator input (PAI) pin is detected. The maximum clocking rate is the system clock divided by four.

In gated time accumulation mode a clock increments PACNT while the PAI pin is in the active state. There are four possible clock sources.

Two bits in the TFLG2 register show pulse accumulator status. The pulse accumulator flag (PAIF) indicates that a selected edge has been detected at the PAI pin. The pulse accumulator overflow flag (PAOVF) indicates that the pulse accumulator count has rolled over from $FF to $00. This can be used to extend the range of the counter beyond 8 bits.

An interrupt request can be made when each of the status flags is set. However, operation of the PAI interrupt depends on operating mode. In event counting mode, an interrupt is requested when the edge being counted is detected. In gated mode, the request is made when the PAI input changes from active to inactive state. Interrupt requests are enabled by the PAOVI and PAII bits in the TMSK2 register.

8

Bits in the pulse accumulator control register (PACTL) control the operation of PACNT. The PAMOD bit selects event counting or gated operation. In event counting mode, the PEDGE control bit determines whether a rising or falling edge is detected; in gated mode, PEDGE specifies the active state of the gate signal. Bits PACLK[1:0] select the clock source used in gated mode.

PACTL and PACNT are implemented as one 16-bit register, but may be accessed with byte or word access cycles. Both registers are cleared at reset, but the PAIS and PCLKS bits show the state of the PAI and PCLK pins.

The PAI pin can also be used for general-purpose input. The logic state of the PAIS bit in PACTL shows the state of the pin.

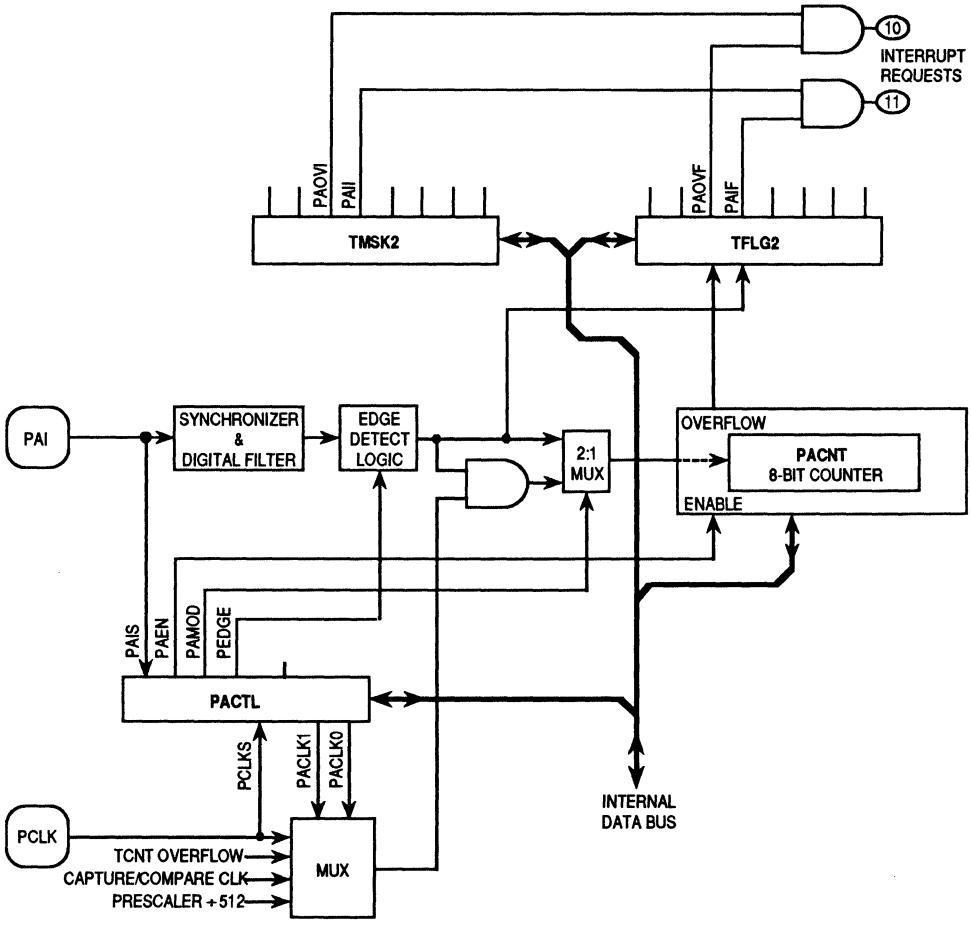


Figure 8–5. Pulse Accumulator Block Diagram

8

8.11 Pulse-Width Modulation Unit

The pulse-width modulation (PWM) unit has two output channels, PWMA and PWMB. A single clock output from the prescaler multiplexer drives a 16-bit counter that is used to control both channels. Figure 8–6 is a block diagram of the pulse-width modulation unit.

The PWM unit has two operational modes. Fast mode uses a clocking rate equal to 1/256 of the prescaler output rate; slow mode uses a rate equal to 1/32768 of the prescaler output rate. The duty cycle ratios of the two PWM channels can be individually controlled by software. The PWMA pin can also output the clock that drives the PWM counter. PWM pins can also be used as output pins.

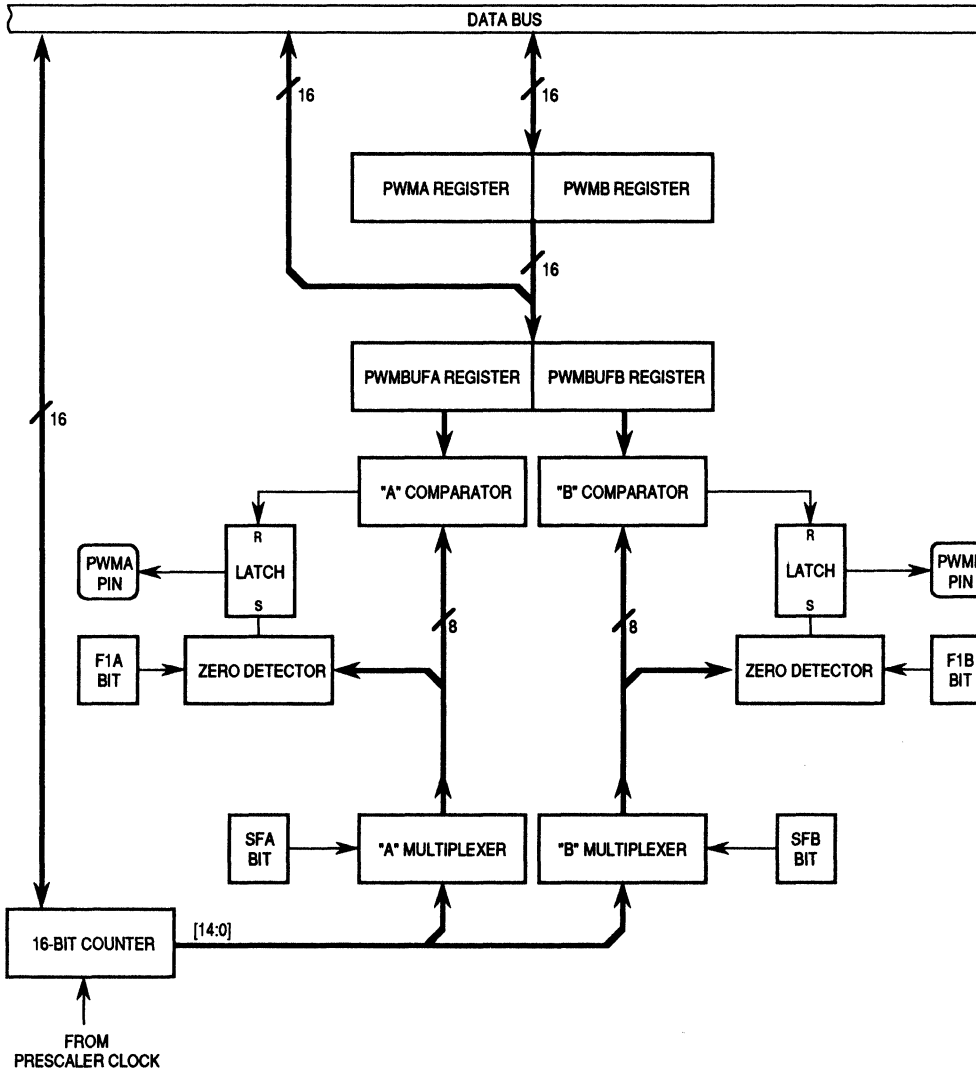


Figure 8–6. PWM Block Diagram

8.11.1 PWM Counter

The 16-bit counter in the PWM unit is similar to the timer counter in the capture/compare unit. During reset, the GPT is configured to use the system clock divided by two to drive the counter. Initialization software can reconfigure the counter to use one of seven prescaler outputs or an external clock input from the PCLK pin.

The PWM count register (PWMCNT can be read at any time without affecting its value. A read must be a word access to ensure coherence, but byte accesses can be made if coherence is not needed. The counter is cleared to $0000 during reset and is a read-only register except in freeze or test mode.

Fifteen of the sixteen counter bits are output to multiplexers A and B. The multiplexers provide the fast and slow modes of the PWM unit. Mode for PWMA is selected by the SFA bit in the PWM control register C (PWMC). Mode for PWMB is selected by the SFB bit in the same register.

PWMA, PWMB, and PPR[2:0] bits in PWMC control PWM output frequency. In fast mode, bits [7:0] of PWMCNT are used to clock the PWM logic; in slow mode, bits [14:7] are used. The period of a PWM output in slow mode is 128 times longer than the fast mode period. Table 8–3 shows a range of PWM output frequencies using a 16.78 MHz system clock.

Table 8–3. PWM Frequency Range Using 16.78 MHz System Clock

PPR[2:0]	Prescaler Tap	Fast Mode	Slow Mode
000	Div 2 = 8.39 MHz	32.8 kHz	256 Hz
001	Div 4 = 4.19 MHz	16.4 kHz	128 Hz
010	Div 8 = 2.10 MHz	8.19 kHz	64.0 Hz
011	Div 16 = 1.05 MHz	4.09 kHz	32.0 Hz
100	Div 32 = 524 kHz	2.05 kHz	16.0 Hz
101	Div 64 = 262 kHz	1.02 kHz	8.0 Hz
110	Div 128 = 131 kHz	512 Hz	4.0 Hz
111	PCLK	PCLK/256	PCLK/32768

8.11.2 PWM Function

The pulse width values of the PWM outputs are determined by control registers PWMA and PWMB. PWMA and PWMB are 8-bit registers that are implemented as two bytes of a 16-bit register. PWMA and PWMB can be accessed as separate bytes or as one 16-bit register. A value of $00 loaded into either register causes the corresponding output pin to output a continuous logic level zero signal. A value of $80 causes the corresponding output signal to have a 50% duty cycle, and so on, to the maximum value of $FF, which corresponds to an output which is at logic level one for 255/256 of the cycle.

Setting the F1A (for PWMA) or F1B (for PWMB) bits in the CFORC register causes the corresponding pin to output a continuous logic level one signal. The logic level of the associated pin does not change until the end of the current cycle. F1A and F1B are the lower two bits of CFORC, but can be accessed at the same word address as PWMC.

Data written to PWMA and PWMB is not used until the end of a complete cycle. This prevents supurious short or long pulses when register values are changed. The current duty cycle value is stored in the appropriate PWM buffer register (PWMBUFA or PWMBUFB). The new value is transferred from the PWM register to the buffer register at the end of the current cycle.

Registers PWMA, PWMB, and PWMC, are reset to $00 during reset. These registers may be written or read at any time. PWMC is implemented as the lower byte of a 16-bit register. The upper byte is the CFORC register. The buffer registers, PWMBUFA and PWMBUFB, are read-only at all times and may be accessed as separate bytes or as one 16-bit register.

Pins PWMA and PWMB can also be used for general-purpose output. The values of the F1A and F1B bits in PWMC are driven out on the corresponding PWM pins when normal PWM operation is disabled. When read, the F1A and F1B bits reflect the states of the PWMA and PWMB pins.

SECTION 9
STANDBY RAM MODULE

The standby RAM (SRAM) module consists of a control register block and a 1-Kbyte array of fast (two bus cycle) static RAM. The SRAM is especially useful for system stacks and variable storage. SRAM can be mapped to any 1 Kbyte boundary in the address map, but must not overlap the module control registers (overlap makes the registers inaccessible). Data can be read/written in bytes, words or long words. SRAM is powered by V_{DD} in normal operation. During power-down, SRAM contents can be maintained by power from the V_{STBY} input. Power switching between sources is automatic.

9.1 SRAM Register Block

There are four SRAM control registers: the RAM module configuration register (RAMMCR), the RAM test register (RAMTST), and the RAM array base address registers (RAMBAH/RAMBAL).

The modmap (MM) bit in the system integration module configuration register (SIMCR) defines the most significant bit (ADDR23) of the IMB address for each module in the MC68HC16Z1. Because the CPU16 in the MC68HC16Z1 drives only ADDR[19:0] and ADDR[23:20] follow the logic state of ADDR19, MM must equal one. **SECTION 4 SYSTEM INTEGRATION MODULE** contains more information about how the state of MM affects the system.

There is an 8-byte minimum control register block size for the SRAM 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.

9.2 SRAM Array Address Mapping

Base address registers RAMBAH and RAMBAL are used to specify the SRAM array base address in the MC68HC16Z1 memory map. RAMBAH and RAMBAL can only be written while the SRAM is in low-power mode (RAMMCR STOP = 1) and the base address lock (RAMMCR RLCK = 0) is disabled. RLCK can be written once only to a value of one. This prevents accidental remapping of the array.

NOTE

In the MC68HC16Z1, ADDR[23:20] follow the logic state of ADDR19 unless externally driven. The SRAM array must not be mapped to addresses $7FF000–$7FFFFF, which are inaccessible to the CPU16. If mapped to these addresses, the array remains inaccessible until a reset occurs.

9.3 SRAM Array Address Space Type

The RASP field in RAMMCR determines SRAM array address space type. The SRAM module can respond to both program and data space accesses or to program space accesses only. This allows code to be executed from RAM, and permits use of program counter relative addressing mode for operand fetches from the array. Refer to **SECTION 4 SYSTEM INTEGRATION MODULE** and **SECTION 5 CENTRAL PROCESSING UNIT** for more information concerning address space types and program/data space access.

9.4 Normal Access

The array can be accessed by byte, word, or long word. A byte or aligned word access takes one bus cycle or two system clocks. A long word or misaligned word access requires two bus cycles. Refer to **SECTION 4 SYSTEM INTEGRATION MODULE** for more information concerning access times.

9.5 Standby and Low-Power Stop Operation

Standby and low-power modes should not be confused. Standby mode maintains the RAM array when the MCU main power supply is turned off. Low-power mode minimizes MCU power consumption.

9

Relative voltage levels of the MCU V_{DD} and V_{STBY} pins determine whether the SRAM is in standby mode. SRAM circuitry senses when the difference between the two supply voltages is greater than a specified limit, and switches to the higher-voltage power source. If specified levels are maintained, there is no loss of memory when switching occurs. Access to the array is not guaranteed while the SRAM module is powered from V_{STBY}. If standby operation is not desired, connect the V_{STBY} pin to the V_{SS} pin.

Setting the STOP bit in RAMMCR switches the SRAM module to low-power mode. In low-power mode, the array retains its contents, but cannot be read or written by the CPU. If V_{DD} falls below V_{SB} while the SRAM is in low-power mode, internal circuitry switches to V_{STBY}, as in standby mode. Because the CPU16 always operates in supervisor mode, STOP can be read or written at any time. STOP is set during reset. Stop mode is exited by clearing STOP.

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

9.6 Reset

Reset places the SRAM in low-power mode, enables program space access, and clears the base address registers and the register lock bit. These actions make it possible to write a new base address into the registers.

When a synchronous reset occurs while a byte or word SRAM 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 RAM may be corrupted by asynchronous reset. Refer to **SECTION 4 SYSTEM INTEGRATION MODULE** for more information concerning resets.

9

APPENDIX A
ELECTRICAL CHARACTERISTICS

This appendix contains electrical specification tables and reference timing diagrams. Each timing diagram has an associated key table made up of parameters abstracted from the specification tables. Pertinent notes have been included in the key tables.

Table A–1. Maximum Ratings

Rating	Symbol	Value	Unit
Supply Voltage[1,2,6]	V_{DD}	–0.3 to + 6.5	V
Input Voltage[1,2,3,4,6]	V_{in}	–0.3 to +6.5	V
Instantaneous Maximum Current Single pin limit (applies to all pins)[1,4,5,6]	I_D	25	mA
Operating Maximum Current Digital input disruptive current [4,5,6,7] $V_{SS} - 0.3 \leq V_{IN} \leq V_{DD} + 0.3$	I_{iD}	– 500 to 500	µA
Operating Temperature Range MC68HC16Z1 "C" Suffix MC68HC16Z1 "V" Suffix MC68HC16Z1 "M" Suffix	T_A	TL to TH –40 to 85 –40 to 105 –40 to 125	°C
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, $\overline{\text{TSTME}}$/TSC, and XFC are internally clamped to V_{DD}.
5. Power supply must maintain regulation within operating V_{DD} range during instantaneous and operating maximum current condition.
6. This parameter is periodically sampled rather than 100% tested.
7. Total input current for all digital input-only and all digital input/output pins must not exceed 10 mA. Exceeding this limit can cause disruption of normal operation.

A

Table A–2. Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal Resistance Plastic 132-Pin Surface Mount Plastic 144-Pin Surface Mount	Θ_{JA}	38	°C/W

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

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

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:

$$P_D = K + (T_J + 273°C) \quad (2)$$

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.

A

Table A–3. 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)

Characteristic	Symbol	Min	Max	Unit
PLL Reference Frequency Range	f_{ref}	25	50	kHz
System Frequency[1]	f_{sys}	dc	16.78	MHz
On-Chip PLL Frequency		0.131	16.78	
External Clock Operation		dc	16	
PLL Lock Time[2]	t_{lpll}	—	20	ms
Limp Mode Clock Frequency[3]	f_{limp}			MHz
SYNCR X bit = 0		—	f_{sys} max /2	
SYNCR X bit = 1		—	f_{sys} max	
CLKOUT Stability[4,5]	C_{stab}			%
Short term		−1.0	1.0	
Long term		−0.5	0.5	

NOTES:

1. All internal registers retain data at 0 Hz.
2. Assumes that stable V_{DDSYN} is applied, that an external filter capacitor with a value of 0.1 µF is attached to the XFC pin, and that the crystal oscillator is stable. Lock time is measured from power-up to $\overline{\text{RESET}}$ release. 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.
3. Determined by the initial control voltage applied to the on-chip VCO. The X bit in SYNCR controls a divide by two scaler on the system clock output.
4. Short-term CLKOUT stability is the average deviation from programmed frequency measured over a 2 µs interval at maximum f_{sys}. Long-term CLKOUT stability is the average deviation from programmed frequency measured over a 1 ms interval at maximum f_{sys}. Stability is measured with a stable external clock applied — variation in crystal oscillator frequency is additive to this figure.
5. This parameter is periodically sampled rather than 100% tested.

A

Table A–4. DC Characteristics

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

Characteristic	Symbol	Min	Max	Unit
Input High Voltage	V_{IH}	0.7 (V_{DD})	V_{DD} + 0.3	V
Input Low Voltage	V_{IL}	V_{SS} – 0.3	0.2 (V_{DD})	V
Input Hysteresis[1,9]	V_{HYS}	0.5	—	V
Input Leakage Current[2] V_{in} = V_{DD} or V_{SS} All input-only pins except ADC pins	I_{in}	–2.5	2.5	µA
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
CMOS Output High Voltage[2,3] I_{OH} = –10.0 µA Group 1,2,4 input/output and all output pins	V_{OH}	V_{DD} – 0.2	—	V
CMOS Output Low Voltage[2] I_{OL} = 10.0 µA Group 1,2,4 input/output and all output pins	V_{OL}	—	0.2	V
Output High Voltage [2,3] I_{OH} = –0.8 mA Group 1,2,4 input/output and all output pins	V_{OH}	V_{DD} – 0.8	—	V
Output Low Voltage[2] I_{OL} = 1.6 mA Group 1 I/O Pins, CLKOUT, FREEZE/QUOT, IPIPE0 I_{OL} = 5.3 mA Group 2 and Group 4 I/O Pins, $\overline{CSBOOT}$, $\overline{BG/CS}$ I_{OL} = 12 mA Group 3	V_{OL}	— — —	0.4 0.4 0.4	V
Three State Control Input High Voltage	V_{IHTSC}	1.6 (V_{DD})	9.1	V
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
V_{DD} Supply Current[6] RUN[4] LPSTOP, 32.768 kHz crystal, VCO Off (STSIM = 0) LPSTOP (External clock input frequency = maximum f_{sys})	I_{DD} S_{IDD} S_{IDD}	— — —	110 350 5	mA µA mA
Clock Synthesizer Operating Voltage	V_{DDSYN}	4.5	5.5	V
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
RAM Standby Voltage[7] Specified V_{DD} applied V_{DD} = V_{SS}	V_{SB}	 0.0 3.0	 5.5 5.5	V
RAM Standby Current[7] Specified V_{DD} applied V_{DD} = V_{SS}	I_{SB} I_{SB}	–2.5 —	2.5 50	µA µA
Power Dissipation[8]	P_D	—	605	mW

A

Table A–4. DC Characteristics (Continued)

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

Characteristic	Symbol	Min	Max	Unit
Input Capacitance[2,9] All input-only pins except ADC pins	C_{in}	—	10	pF
All input/output pins		—	20	
Load Capacitance[2]				
Group 1 I/O Pins and CLKOUT, FREEZE/QUOT, IPIPE0	C_L	—	90	pF
Group 2 I/O Pins and $\overline{CSBOOT}$, BG/$\overline{CS}$		—	100	
Group 3 I/O pins		—	130	
Group 4 I/O pins		—	200	

NOTES:

1. Applies to:
   Port ADA [7:0] — AN[7:0]
   Port E [7:4] — SIZ[1:0], $\overline{AS}$, $\overline{DS}$
   Port F [7:0] — $\overline{IRQ}$[7:1], MODCLK
   Port GP [7:0] — IC4/OC5/OC1, IC[3:1], OC[4:1]/OC1
   Port QS [7:0] — TXD, PCS[3:1], PCS0/$\overline{SS}$, SCK, MOSI, MISO
   $\overline{BKPT}$/DSCLK, DSI/IPIPE1, PAI, PCLK, $\overline{RESET}$, RXD, $\overline{TSTME}$/TSC
2. Input-Only Pins: $\overline{TSTME}$/TSC, $\overline{BKPT}$/DSCLK, PAI, PCLK, RXD
   Output-Only Pins: $\overline{CSBOOT}$, BG/$\overline{CS1}$, CLKOUT, FREEZE/QUOT, DS0/IPIPE0, PWMA, PWMB
   Input/Output Pins:
   Group 1: Port GP [7:0] — IC4/OC5/OC1, IC[3:1], OC[4:1]/OC1
   DATA[15:0], DSI/IPIPE1
   Group 2: Port C [6:0] — ADDR[22:19]/$\overline{CS[9:6]}$, FC[2:0]/$\overline{CS[5:3]}$
   Port E [7:0] — SIZ[1:0], $\overline{AS}$, $\overline{DS}$, $\overline{AVEC}$, DSACK[1:0]
   Port F [7:0] — $\overline{IRQ}$[7:1], MODCLK
   Port QS [7:3] — TXD, PCS[3:1], PCS0/$\overline{SS}$
   ADDR23/$\overline{CS10}$/ECLK, ADDR[18:0], R/$\overline{W}$, $\overline{BERR}$, $\overline{BR}$/$\overline{CS0}$, $\overline{BGACK}$/$\overline{CS2}$
   Group 3: $\overline{HALT}$, $\overline{RESET}$
   Group 4: MISO, MOSI, SCK
3. Does not apply to $\overline{HALT}$ and $\overline{RESET}$ because they are open drain pins. Does not apply to Port QS [7:0] (TXD, PCS[3:1], PCS0/$\overline{SS}$, 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 V_{DD} supply and V_{DDSYN} supply currents.
7. The SRAM module will not switch into standby mode as long as V_{SB} does not exceed V_{DD} by more than 0.5 Volt. The SRAM array cannot be accessed while the module is in standby mode.
8. Power dissipation measured with system clock frequency of 16.78 MHz, all modules active. Power dissipation is calculated using the following expression:
   $$P_D = \text{Maximum } V_{DD} (I_{DDSYN} + I_{DD})$$
9. This parameter is periodically sampled rather than 100% tested.

A

Table A–5. 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[2]	Frequency of Operation (32.768 kHz crystal)	f	0.13	16.78	MHz
1	Clock Period	t_{cyc}	59.6	—	ns
1A	ECLK Period	t_{Ecyc}	476	—	ns
1B[3]	External Clock Input Period	t_{Xcyc}	64	—	ns
2, 3	Clock Pulse Width	t_{CW}	24	—	ns
2A, 3A	ECLK Pulse Width	t_{ECW}	236	—	ns
2B, 3B[3]	External Clock Input High/Low Time	t_{XCHL}	32	—	ns
4, 5	CLKOUT 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 Input Rise and Fall Time	t_{XCrf}	—	5	ns
6	Clock High to ADDR, FC, SIZE Valid	t_{CHAV}	0	29	ns
7	Clock High to ADDR, Data, FC, SIZE, High Impedance	t_{CHAZx}	0	59	ns
8	Clock High to ADDR, FC, SIZE, Invalid	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Asserted	t_{CLSA}	2	25	ns
9A[4]	$\overline{AS}$ to $\overline{DS}$ or $\overline{CS}$ Asserted (Read)	t_{STSA}	–15	15	ns
11	ADDR, FC, SIZE Valid to $\overline{AS}$, $\overline{CS}$, (and $\overline{DS}$ Read) Asserted	t_{AVSA}	15	—	ns
12	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated	t_{CLSN}	2	29	ns
13	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to ADDR, FC, SIZE Invalid (Address Hold)	t_{SNAI}	15	—	ns
14	$\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Width Asserted	t_{SWA}	100	—	ns
14A	$\overline{DS}$, $\overline{CS}$ Width Asserted (Write)	t_{SWAW}	45	—	ns
14B	$\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Width Asserted (Fast Cycle)	t_{SWDW}	40	—	ns
15[5]	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Width Negated	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/$\overline{W}$ High	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/$\overline{W}$ High to $\overline{AS}$, $\overline{CS}$ Asserted	t_{RAAA}	15	—	ns

A

Table A–5. 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
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 $\overline{AS}$, $\overline{CS}$ (Fast Write Cycle)	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[6]	$\overline{DS}$, $\overline{CS}$ Negated to Data In Invalid (Data In Hold)	t_{SNDI}	0	—	ns
29A[6, 7]	$\overline{DS}$, $\overline{CS}$ Negated to Data In High Impedance	t_{SHDI}	—	55	ns
30[6]	CLKOUT Low to Data In Invalid (Fast Cycle Hold)	t_{CLDI}	15	—	ns
30A[6]	CLKOUT Low to Data In High Impedance	t_{CLDH}	—	90	ns
31[8]	$\overline{DSACK[1:0]}$ Asserted to Data In Valid	t_{DADI}	—	50	ns
33	Clock Low to $\overline{BG}$ Asserted/Negated	t_{CLBAN}	—	29	ns
35[9]	$\overline{BR}$ Asserted to $\overline{BG}$ Asserted	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[10]	$\overline{DSACK[1:0]}$ Asserted to $\overline{BERR}$, $\overline{HALT}$ Asserted	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
70	Clock Low to Data Bus Driven (Show Cycle)	t_{SCLDD}	0	29	ns
71	Data Setup Time to Clock Low (Show Cycle)	t_{SCLDS}	15	—	ns
72	Data Hold from Clock Low (Show Cycle)	t_{SCLDH}	10	—	ns
73	$\overline{BKPT}$ Input Setup Time	t_{BKST}	15	—	ns
74	$\overline{BKPT}$ Input Hold Time	t_{BKHT}	10	—	ns

A

Table A–5. 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
75	Mode Select Setup Time	t_{MSS}	20	—	t_{cyc}
76	Mode Select Hold Time	t_{MSH}	0	—	ns
77	$\overline{RESET}$ Assertion Time[11]	t_{RSTA}	4	—	t_{cyc}
78	$\overline{RESET}$ Rise Time[12]	t_{RSTR}	—	10	t_{cyc}
100	CLKOUT High to Phase 1 Asserted[13]	t_{CHP1A}	3	40	ns
101	CLKOUT High to Phase 2 Asserted[13]	t_{CHP2A}	3	40	ns
102	Phase 1 Valid to $\overline{AS}$ or $\overline{DS}$ Asserted[13]	t_{P1VSA}	10	—	ns
103	Phase 2 Valid to $\overline{AS}$ or $\overline{DS}$ Negated[13]	t_{P2VSN}	10	—	ns
104	$\overline{AS}$ or $\overline{DS}$ Valid to Phase 1 Negated[13]	t_{SAP1N}	10	—	ns
105	$\overline{AS}$ or $\overline{DS}$ Negated to Phase 2 Negated[13]	t_{SNP2N}	10	—	ns

NOTES:

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. Minimum external clock high and low times are based on a 50% duty cycle. The minimum allowable t_{Xcyc} period will be 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).

   To achieve maximum operating frequency (f_{sys}) while using an external clock input, adjust clock input duty cycle to obtain a 50% duty cycle on CLKOUT.
4. 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.
5. 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.
6. 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.
7. Maximum value is equal to (t_{cyc} / 2) + 25 ns.
8. 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.
9. To ensure coherency during every operand transfer, $\overline{BG}$ is not asserted in response to $\overline{BR}$ until after all cycles of the current operand transfer are complete.
10. In the absence of $\overline{DSACK[1:0]}$, $\overline{BERR}$ is an asynchronous input using the asynchronous setup time (specification 47A).
11. 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}
12. External logic must pull $\overline{RESET}$ high during this period in order for normal MCU operation to begin.
13. Eight pipeline states are multiplexed into IPIPE[1:0]. The multiplexed signals have two phases.
14. 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.

Table A–6. 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 IPIPE1 High Impedance	t_{IFZ}	—	50	ns
B8	CLKOUT High to IPIPE1 Valid	t_{IF}	—	50	ns
B9	DSCLK Low Time	t_{DSCLO}	1	—	t_{cyc}

NOTES:

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

Table A–7. 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[2]	ECLK Low to Address Valid	t_{EAD}	—	60	ns
E2	ECLK Low to Address Hold	t_{EAH}	10	—	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}	15	—	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}	5	—	ns
E13	$\overline{CS}$ Negated to Data Hold (Write)	t_{ECHW}	0	—	ns
E14[3]	Address Access Time (Read)	t_{EACC}	386	—	ns
E15[4]	Chip Select Access Time (Read)	t_{EACS}	296	—	ns
E16	Address Setup Time	t_{EAS}	—	1/2	t_{cyc}

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
2. When 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 = $t_{Ecyc} - t_{ECSD} - t_{EDSR}$

A

Table A–8. 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 Output	 t_{ri} t_{ro}	 — —	 2 30	 µs ns
13	Fall Time Input 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

A

Table A–9. ADC Maximum Ratings

Num	Parameter	Symbol	Min	Max	Unit
1	Analog Supply	V_{DDA}	– 0.3	6.5	V
2	Internal Digital Supply	V_{DDI}	– 0.3	6.5	V
3	Reference Supply	V_{RH}, V_{RL}	– 0.3	6.5	V
4	V_{SS} Differential Voltage	$V_{SSI} - V_{SSA}$	– 0.1	0.1	V
5	V_{DD} Differential Voltage	$V_{DDI} - V_{DDA}$	– 6.5	6.5	V
6	V_{REF} Differential Voltage	$V_{RH} - V_{RL}$	– 6.5	6.5	V
7	V_{REF} to V_{DDA} Differential Voltage	$V_{RH} - V_{DDA}$	– 6.5	6.5	V
8	Disruptive Input Current[1, 2, 4, 5] $V_{SSA} - 0.3 \le V_{INA} \le V_{DDA} + 2$	I_{NA}	– 15	15	µA
9	Maximum Input Current[3, 4] $V_{SSA} - 1 \le V_{INA} \le V_{DDA} + 3.5$	I_{MA}	– 500	500	µA

NOTES:

1. Below disruptive current conditions, the channel being stressed will have conversion values of $3FF for analog inputs greater than V_{RH} and $000 for values less than V_{RL}. This assumes that $V_{RH} \le V_{DDA}$ and $V_{RL} \ge V_{SSA}$ due to the presence of the sample amplifier. Other channels are not affected by non-disruptive conditions.
2. Input signals with large slew rates or high frequency noise components cannot be converted accurately. These signals also interfere with conversion of other channels.
3. Exceeding limit may cause conversion error on stressed channels and on unstressed channels. Transitions within the limit do not affect device reliability or cause permanent damage.
4. This parameter is periodically sampled rather than 100% tested.
5. Applies to single pin only.

Table A–10. ADC DC Electrical Characteristics (Operating)

(V_{SS} = 0 Vdc, ADCLK = 2.1 MHz, T_A within operating temperature range)

Num	Parameter	Symbol	Min	Max	Unit
1	Analog Supply[1]	V_{DDA}	4.5	5.5	V
2	Internal Digital Supply[1]	V_{DDI}	4.5	5.5	V
3	V_{SS} Differential Voltage	$V_{SSI} - V_{SSA}$	– 1.0	1.0	mV
4	V_{DD} Differential Voltage	$V_{DDI} - V_{DDA}$	– 1.0	1.0	V
5	Reference Voltage Low[2, 5]	V_{RL}	V_{SSA}	$V_{DDA} / 2$	V
6	Reference Voltage High[2, 5]	V_{RH}	$V_{DDA} / 2$	V_{DDA}	V
7	V_{REF} Differential Voltage[5]	$V_{RH} - V_{RL}$	4.5	5.5	V
8	Input Voltage[2]	V_{INDC}	V_{SSA}	V_{DDA}	V
9	Input High, Port ADA	V_{IH}	0.7 (V_{DDA})	V_{DDA} + 0.3	V
10	Input Low, Port ADA	V_{IL}	V_{SSA} – 0.3	0.2 (V_{DDA})	V
15	Analog Supply Current[3]	I_{DDA}	—	1.0	mA
16	Analog Supply Current, LPSTOP	S_{DDA}	—	TBD	µA
17	Reference Supply Current	I_{REF}	—	250	µA
18	Input Current, Off Channel[4]	I_{OFF}	—	250	nA
19	Total Input Capacitance, Not Sampling	C_{INN}	—	10	pF
20	Total Input Capacitance, Sampling	C_{INS}	—	15	pF

NOTES:

1. Refers to operation over full temperature and frequency range.
2. To obtain full-scale, full-range results, $V_{SSA} \le V_{RL} \le V_{INDC} \le V_{RH} \le V_{DDA}$.
3. Current measured at maximum system clock frequency with ADC active.
4. Maximum leakage occurs at maximum operating temperature. Current decreases by approximately one-half for each 10° C decrease from maximum temperature.
5. Accuracy tested and guaranteed at $V_{RH} - V_{RL} \le 5.0\ V \pm 10\%$.

A

Table A–11. ADC AC Characteristics (Operating)

(V_{DD} and V_{DDA} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A within operating temperature range)

Num	Parameter	Symbol	Min	Max	Unit
1	IMB Clock Frequency	F_{ICLK}	2.0	16.78	MHz
2	ADC Clock Frequency	F_{ADCLK}	0.5	2.1	MHz
3	8-bit Conversion Time (16 ADC Clocks)[1]	T_{CONV}	7.62	—	µs
4	10-bit Conversion Time (18 ADC Clocks)[1]	T_{CONV}	8.58	—	µs
5	Stop Recovery Time	T_{SR}	—	10	µs

NOTES:

1. Assumes 2.1 MHz ADC clock and selection of minimum sample time (2 ADC clocks).

Table A–12. ADC Conversion Characteristics (Operating)

(V_{DD} and V_{DDA} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, ADCLK = 2.1 MHz)

Num	Parameter	Symbol	Min	Typ	Max	Unit
1	8-bit Resolution[1]	1 Count	—	20	—	mV
2	8-bit Differential Nonlinearity[2]	DNL	–.5	—	.5	Counts
3	8-bit Integral Nonlinearity[2]	INL	–1	—	1	Counts
4	8-bit Absolute Error[2,3]	AE	–1	—	1	Counts
5	10-bit Resolution[1]	1 Count	—	5	—	mV
6	10-bit Differential Nonlinearity[2]	DNL	–1	—	1	Counts
7	10-bit Integral Nonlinearity[2]	INL	–2	—	2	Counts
8	10-bit Absolute Error[2,4]	AE	–2.5	—	2.5	Counts
9	Source Impedance at Input[5]	R_S	—	20	See Note 5	kΩ

NOTES:

1. $V_{RH} - V_{RL} \geq 5.12$ V; $V_{DDA} - V_{SSA} = 5.12$ V
2. At $V_{REF} = 5.12$ V, one 10-bit count = 5 mV and one 8-bit count = 20 mV.
3. 8-bit absolute error of 1 count (20 mV) includes 1/2 count (10 mV) inherent quantization error and 1/2 count (10 mV) circuit (differential, integral, and offset) error.
4. 10-bit absolute error of 2.5 counts (12.5 mV) includes 1/2 count (2.5 mV) inherent quantization error and 2 counts (10 mV) circuit (differential, integral, and offset) error.
5. Maximum source impedance is application-dependent. Error resulting from pin leakage depends on junction leakage into the pin and on leakage due to charge-sharing with internal capacitance. In the following expressions, expected error in result value due to leakage is expressed in voltage (V_{errx}).

   Error from junction leakage is a function of external source impedance and input leakage current:

   $$V_{errj} = R_S \times I_{OFF}$$

   where I_{OFF} is a function of operating temperature. (See Table A–10, note 4).

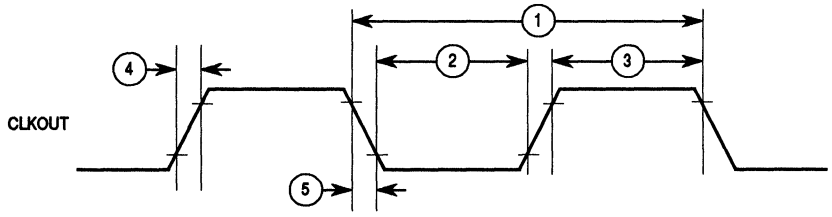
   Charge-sharing leakage is a function of ADC clock speed, number of channels scanned, and source impedance:

   For 10-bit conversion, $V_{err10} = .25\ pF \times V_{DDA} \times R_S \times ADCLK + (9 \times \text{number of channels})$

   For 8-bit conversion, $V_{err8} = .25\ pF \times V_{DDA} \times R_S \times ADCLK + (8 \times \text{number of channels})$

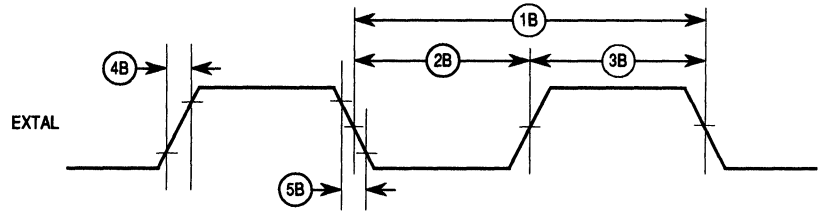
A

Timing Diagrams



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

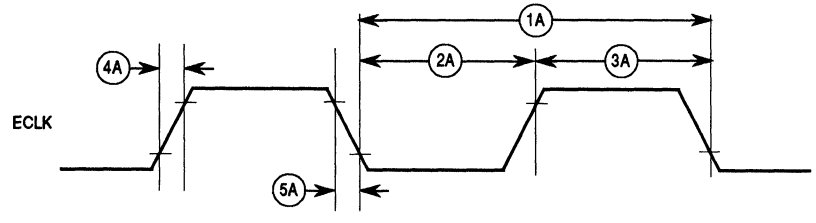
Figure A–1. CLKOUT Output Timing Diagram



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

A



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

Figure A–3. ECLK Output Timing Diagram

Key to Figures A–1, A–2, A–3

(Abstracted from Table A–5; see table for complete notes)

Num	Characteristic	Symbol	Min	Max	Units
1	Clock Period	t_{cyc}	59.6	—	ns
1A	ECLK Period	t_{Ecyc}	476	—	ns
1B[3]	External Clock Input Period	t_{Xcyc}	64	—	ns
2, 3	Clock Pulse Width	t_{CW}	24	—	ns
2A, 3A	ECLK Pulse Width	t_{ECW}	236	—	ns
2B,3B[3]	External Clock Input High/Low Time	t_{XCHL}	32	—	ns
4, 5	CLOCKOUT 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	t_{XCrf}	—	5	ns

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
3. Minimum external clock high and low times are based on a 50% duty cycle. The minimum allowable t_{XCYC} period will be 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).

   To achieve maximum operating frequency (f_{sys}) while using an external clock input, adjust clock input duty cycle to obtain a 50% duty cycle on CLKOUT.

A

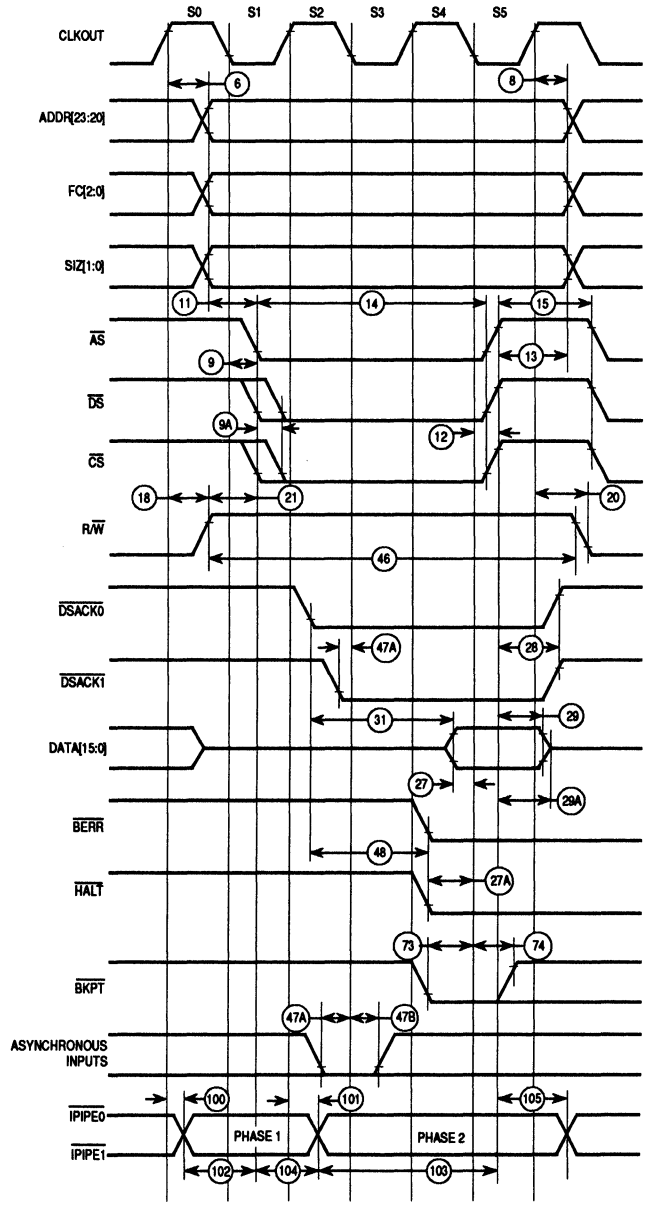


Figure A–4. Read Cycle Timing Diagram

Key to Figure A–4

(Abstracted from Table A–5; see table for complete notes)

Num	Characteristic	Symbol	Min	Max	Units
6	Clock High to ADDR, FC, SIZE Valid	t_{CHAV}	0	29	ns
8	Clock High to ADDR, FC, SIZE Invalid	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Asserted	t_{CLSA}	2	25	ns
9A[4]	$\overline{AS}$ to $\overline{DS}$ or $\overline{CS}$ Asserted (Read)	t_{STSA}	–15	15	ns
11	ADDR, FC, SIZE Valid to $\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Asserted	t_{AVSA}	15	—	ns
12	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated	t_{CLSN}	2	29	ns
13	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to ADDR, FC, SIZE Invalid (Address Hold)	t_{SNAI}	15	—	ns
14	$\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Width Asserted	t_{SWA}	100	—	ns
15[5]	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Width Negated	t_{SN}	40	—	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/$\overline{W}$ High to $\overline{AS}$, $\overline{CS}$ Asserted	t_{RAAA}	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[6]	$\overline{DS}$, $\overline{CS}$ Negated to Data In Invalid (Data In Hold)	t_{SNDI}	0	—	ns
29A[6,7]	$\overline{DS}$, $\overline{CS}$ Negated to Data In High-Impedance	t_{SHDI}	—	55	ns
31[8]	$\overline{DSACK[1:0]}$ Asserted to Data In Valid	t_{DADI}	—	50	ns
46	R/$\overline{W}$ Width Asserted (Write or Read)	t_{RWA}	150	—	ns
47A	Asynchronous Input Setup Time $\overline{BR}$, $\overline{BG}$, $\overline{DSACK[1:0]}$, $\overline{BERR}$, $\overline{AVEC}$, $\overline{HALT}$	t_{AIST}	5	—	ns
47B	Asynchronous Input Hold Time	t_{AIHT}	15	—	ns
48[10]	$\overline{DSACK[1:0]}$ Asserted to $\overline{BERR}$, $\overline{HALT}$ Asserted	t_{DABA}	—	30	ns
73	$\overline{BKPT}$ Input Setup Time	t_{BKST}	15	—	ns
74	$\overline{BKPT}$ Input Hold Time	t_{BKHT}	10	—	ns
100	CLKOUT High to Phase 1 Asserted[12]	t_{CHP1A}	3	40	ns
101	CLKOUT High to Phase 2 Asserted[12]	t_{CHP2A}	3	40	ns
102	Phase 1 Valid to $\overline{AS}$ or $\overline{DS}$ Asserted[13]	t_{P1VSA}	10	—	ns
103	Phase 2 Valid to $\overline{AS}$ or $\overline{DS}$ Negated[13]	t_{P2VSN}	10	—	ns
104	$\overline{AS}$ or $\overline{DS}$ Valid to Phase 1 Negated[13]	t_{SAP1N}	10	—	ns
105	$\overline{AS}$ or $\overline{DS}$ Negated to Phase 2 Negated[13]	t_{SNP2N}	10	—	ns

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
4. 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.
5. 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 synchronized to ECLK.
6. Hold times are specified with respect to $\overline{DS}$ or $\overline{CS}$ on asynchronous reads and with respect to CLKOUT on fast reads. The user is free to use either hold time.
7. Maximum value is equal to (t_{cyc} / 2) + 25 ns.
8. 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. In the absence of $\overline{DSACK[1:0]}$, $\overline{BERR}$ is an asynchronous input — use specification 47A.
13. Eight pipeline states are multiplexed into IPIPE[1:0]. The multiplexed signals have two phases.

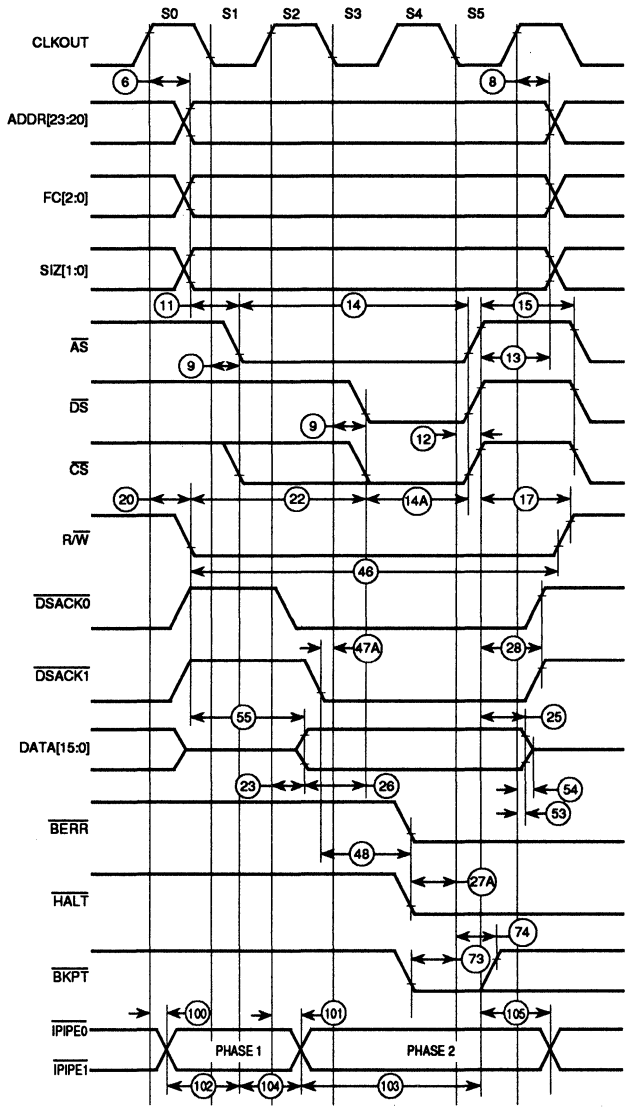


Figure A–5. Write Cycle Timing Diagram

Key to Figure A–5

(Abstracted from Table A–5; see table for complete notes)

Num	Characteristic	Symbol	Min	Max	Units
6	Clock High to ADDR, FC, SIZE Valid	t_{CHAV}	0	29	ns
8	Clock High to ADDR, FC, SIZE Invalid	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Asserted	t_{CLSA}	2	25	ns
11	ADDR, FC, SIZE, Valid to $\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Asserted	t_{AVSA}	15	—	ns
12	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated	t_{CLSN}	2	29	ns
13	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to ADDR, FC, SIZE Invalid (Address Hold)	t_{SNAI}	15	—	ns
14	$\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Width Asserted	t_{SWA}	100	—	ns
14A	$\overline{DS}$, $\overline{CS}$ Width Asserted Write	t_{SWAW}	45	—	ns
15[5]	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Width Negated	t_{SN}	40	—	ns
17	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to R/$\overline{W}$ High	t_{SNRN}	15	—	ns
20	Clock High to R/$\overline{W}$ Low	t_{CHRL}	0	29	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
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
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
46	R/$\overline{W}$ Width Asserted (Write or Read)	t_{RWA}	150	—	ns
47A	Asynchronous Input Setup Time $\overline{BR}$, $\overline{BG}$, $\overline{DSACK[1:0]}$, $\overline{BERR}$, $\overline{AVEC}$, $\overline{HALT}$	t_{AIST}	5	—	ns
48[10]	$\overline{DSACK[1:0]}$ Asserted to $\overline{BERR}$, $\overline{HALT}$ Asserted	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
73	$\overline{BKPT}$ Input Setup Time	t_{BKST}	15	—	ns
74	$\overline{BKPT}$ Input Hold Time	t_{BKHT}	10	—	ns
100	CLKOUT High to Phase 1 Asserted[12]	t_{CHP1A}	3	40	ns
101	CLKOUT High to Phase 2 Asserted[12]	t_{CHP2A}	3	40	ns
102	Phase 1 Valid to $\overline{AS}$ or $\overline{DS}$ Asserted[13]	t_{P1VSA}	10	—	ns
103	Phase 2 Valid to $\overline{AS}$ or $\overline{DS}$ Negated[13]	t_{P2VSN}	10	—	ns
104	$\overline{AS}$ or $\overline{DS}$ Valid to Phase 1 Negated[13]	t_{SAP1N}	10	—	ns
105	$\overline{AS}$ or $\overline{DS}$ Negated to Phase 2 Negated[13]	t_{SNP2N}	10	—	ns

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
5. 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 sychronized to ECLK.
10. In the absence of $\overline{DSACK[1:0]}$, $\overline{BERR}$ is an asynchronous input — use specification 47A.
13. Eight pipeline states are multiplexed into IPIPE[1:0]. The multiplexed signals have two phases.

A

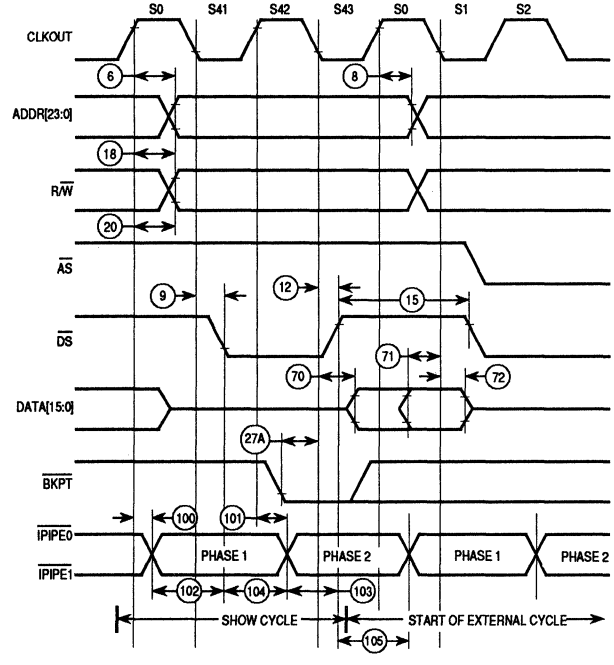


Figure A–6. Show Cycle Timing Diagram

A

Key to Figure A–6

(Abstracted from Table A–5; see table for complete notes)

Num	Characteristic	Symbol	Min	Max	Units
6	Clock High to ADDR, FC, SIZE Valid	t_{CHAV}	0	29	ns
8	Clock High to ADDR, FC, SIZE Invalid	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Asserted	t_{CLSA}	2	25	ns
12	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated	t_{CLSN}	2	29	ns
15[5]	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Width Negated	t_{SN}	40	—	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
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
100	CLKOUT High to Phase 1 Asserted[12]	t_{CHP1A}	3	40	ns
101	CLKOUT High to Phase 2 Asserted[12]	t_{CHP2A}	3	40	ns
102	Phase 1 Valid to $\overline{AS}$ or $\overline{DS}$ Asserted[13]	t_{P1VSA}	10	—	ns
103	Phase 2 Valid to $\overline{AS}$ or $\overline{DS}$ Negated[13]	t_{P2VSN}	10	—	ns
104	$\overline{AS}$ or $\overline{DS}$ Valid to Phase 1 Negated[13]	t_{SAP1N}	10	—	ns
105	$\overline{AS}$ or $\overline{DS}$ Negated to Phase 2 Negated[13]	t_{SNP2N}	10	—	ns

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
5. 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 synchronized to ECLK.
13. Eight pipeline states are multiplexed into IPIPE[1:0]. The multiplexed signals have two phases.

A

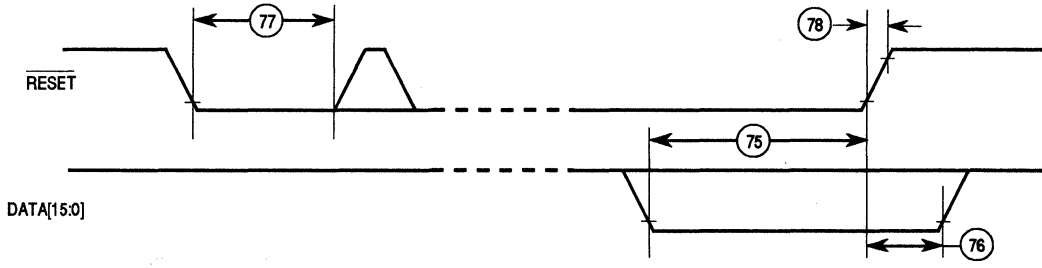


Figure A–7. Reset and Mode Select Timing Diagram

A

Key to Figure A–7

(Abstracted from Table A–5; see table for complete notes)

Num	Characteristic	Symbol	Min	Max	Units
75	Mode Select Setup Time	t_{MSS}	20	—	t_{cyc}
76	Mode Select Hold Time	t_{MSH}	0	—	ns
77	$\overline{\text{RESET}}$ Assertion Time[11]	t_{RSTA}	4	—	t_{cyc}
78	$\overline{\text{RESET}}$ Rise Time[12]	t_{RSTR}	—	10	t_{cyc}

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
11. After external $\overline{\text{RESET}}$ negation is detected, a short transition period (approximately 2 t_{cyc}) elapses, then the SIM drives $\overline{\text{RESET}}$ low for 512 t_{cyc}
12. External logic must pull $\overline{\text{RESET}}$ high during this period in order for normal MCU operation to begin.

A

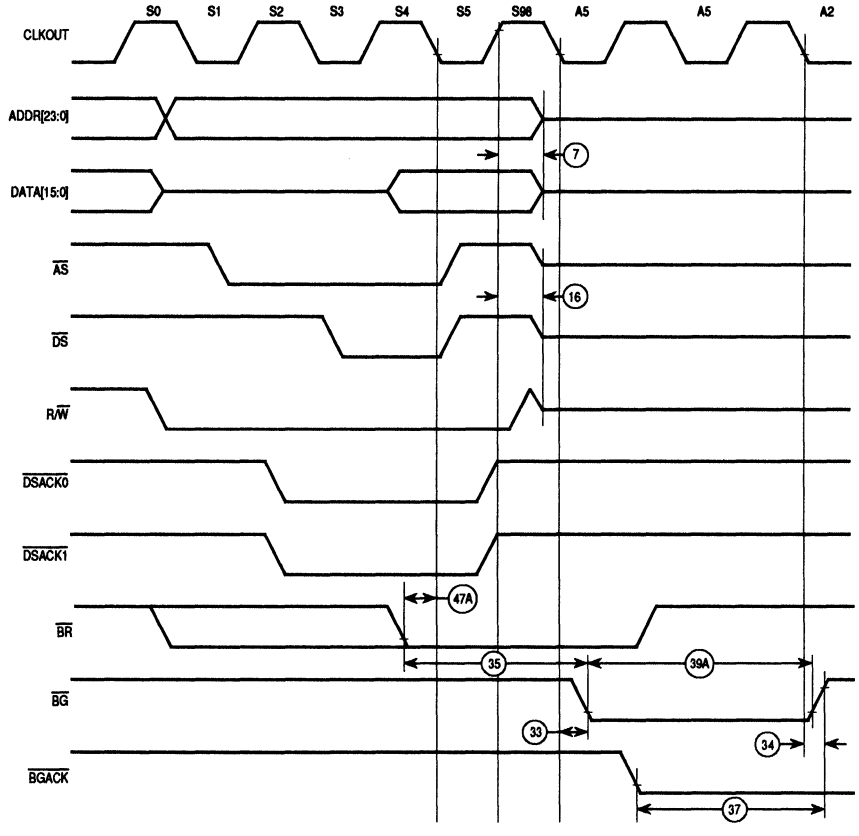


Figure A–8. Bus Arbitration Timing Diagram — Active Bus Case

Key to Figure A–8
(Abstracted from Table A–5; see table for complete notes)

Num	Characteristic	Symbol	Min	Max	Units
7	Clock High to ADDR, Data, FC, SIZE High Impedance	t_{CHAZx}	0	59	ns
16	Clock High to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ High Impedance	t_{CHSZ}	—	59	ns
33	Clock Low to $\overline{BG}$ Asserted/Negated	t_{CLBAN}	—	29	ns
35[9]	$\overline{BR}$ Asserted to $\overline{BG}$ Asserted	t_{BRAGA}	1	—	t_{cyc}
37	$\overline{BGACK}$ Asserted to $\overline{BG}$ Negated	t_{GAGN}	1	2	t_{cyc}
39A	$\overline{BG}$ Width Asserted	t_{GA}	1	—	t_{cyc}
47A	Asynchronous Input Setup Time $\overline{BR}$, $\overline{BG}$, $\overline{DSACK[1:0]}$, $\overline{BERR}$, $\overline{AVEC}$, $\overline{HALT}$	t_{AIST}	5	—	ns

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
9. 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.
13. Eight pipeline states are multiplexed into IPIPE[1:0]. The multiplexed signals have two phases.

A

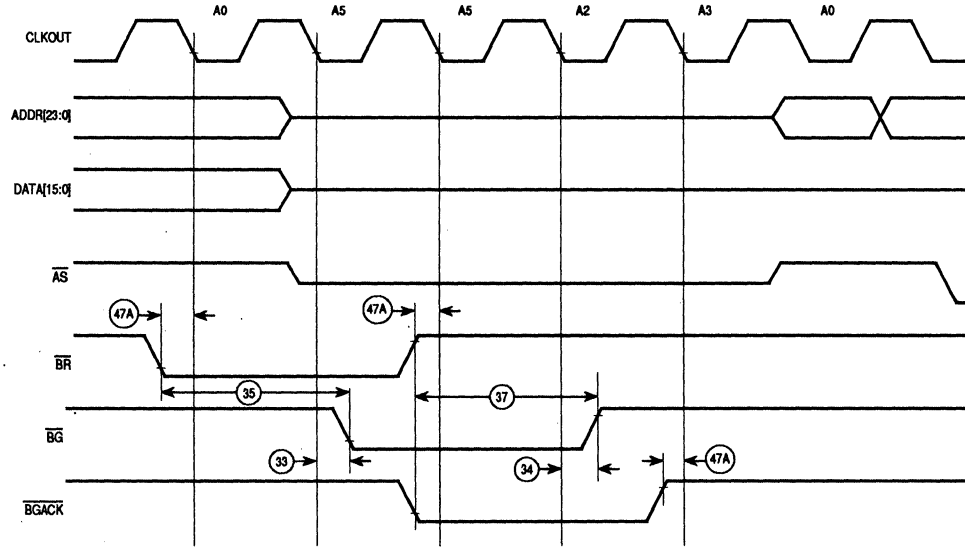


Figure A–9. Bus Arbitration Timing Diagram — Idle Bus Case

Key to Figure A–9

(Abstracted from Table A–5; see table for complete notes)

Num	Characteristic	Symbol	Min	Max	Units
33	Clock Low to $\overline{BG}$ Asserted/Negated	t_{CLBAN}	—	29	ns
35[9]	$\overline{BR}$ Asserted to $\overline{BG}$ Asserted	t_{BRAGA}	1	—	t_{cyc}
37	$\overline{BGACK}$ Asserted to $\overline{BG}$ Negated	t_{GAGN}	1	2	t_{cyc}
47A	Asynchronous Input Setup Time $\overline{BR}$, $\overline{BG}$, $\overline{DSACK[1:0]x}$, $\overline{BERR}$, $\overline{AVEC}$, $\overline{HALT}$	t_{AIST}	5	—	ns

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
9. 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.
13. Eight pipeline states are multiplexed into IPIPE[1:0]. The multiplexed signals have two phases.

A

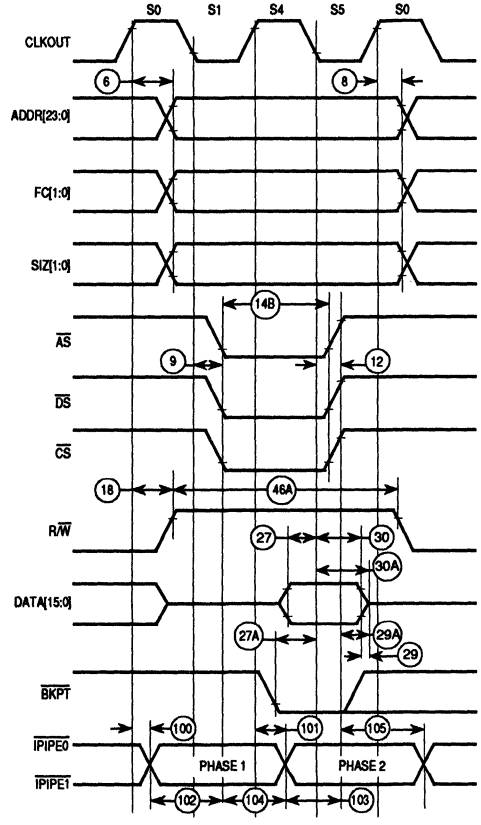


Figure A–10. Fast Termination Read Cycle Timing Diagram

Key to Figure A–10

(Abstracted from Table A–5; see table for complete notes)

Num	Characteristic	Symbol	Min	Max	Units
6	Clock High to ADDR, FC, SIZE Valid	t_{CHAV}	0	29	ns
8	Clock High to ADDR, FC, SIZE Invalid	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Asserted	t_{CLSA}	2	25	ns
12	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated	t_{CLSN}	2	29	ns
14B	$\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Width Asserted (Fast Cycle)	t_{SWDW}	40	—	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
27	Data In Valid to Clock Low (Data Setup)	t_{DICL}	5	—	ns
29[6]	$\overline{DS}$, $\overline{CS}$ Negated to Data In Invalid (Data In Hold)	t_{SNDI}	0	—	ns
29A[6, 7]	$\overline{DS}$, $\overline{CS}$ Negated to Data In High Impedance	t_{SHDI}	—	55	ns
30[6]	CLKOUT Low to Data In Invalid (Fast Hold)	t_{CLDI}	15	—	ns
30A[6]	CLKOUT Low to Data In High-Impedance	t_{CLDH}	—	90	ns
46A	R/$\overline{W}$ Width Asserted (Fast Write or Read)	t_{RWAS}	90	—	ns
73	$\overline{BKPT}$ Input Setup Time	t_{BKST}	15	—	ns
74	$\overline{BKPT}$ Input Hold Time	t_{BKHT}	10	—	ns
100	CLKOUT High to Phase 1 Asserted[12]	t_{CHP1A}	3	40	ns
101	CLKOUT High to Phase 2 Asserted[12]	t_{CHP2A}	3	40	ns
102	Phase 1 Valid to $\overline{AS}$ or $\overline{DS}$ Asserted[13]	t_{P1VSA}	10	—	ns
103	Phase 2 Valid to $\overline{AS}$ or $\overline{DS}$ Negated[13]	t_{P2VSN}	10	—	ns
104	$\overline{AS}$ or $\overline{DS}$ Valid to Phase 1 Negated[13]	t_{SAP1N}	10	—	ns
105	$\overline{AS}$ or $\overline{DS}$ Negated to Phase 2 Negated[13]	t_{SNP2N}	10	—	ns

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
6. Hold times are specified with respect to $\overline{DS}$ or $\overline{CS}$ on asynchronous reads and with respect to CLKOUT on fast reads. The user is free to use either hold time.
7. Maximum value is equal to $(t_{cyc} / 2) + 25$ ns.
13. Eight pipeline states are multiplexed into IPIPE[1:0]. The multiplexed signals have two phases.

A

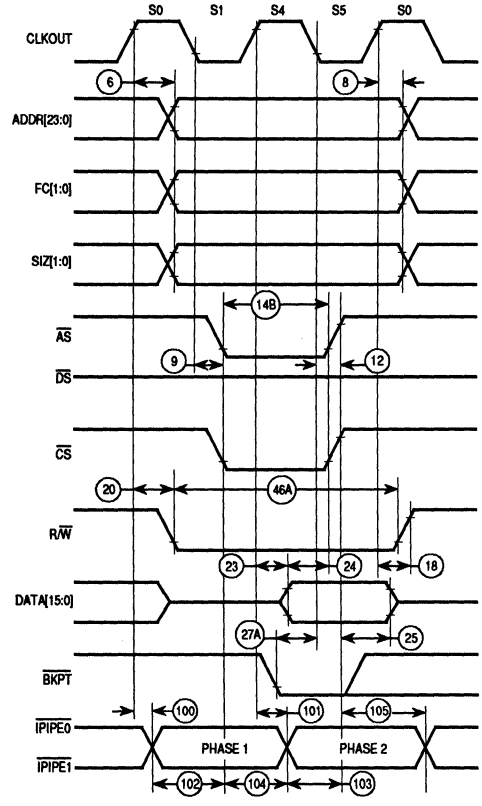


Figure A–11. Fast Termination Write Cycle Timing Diagram

Key to Figure A–11

(Abstracted from Table A–5; see table for complete notes)

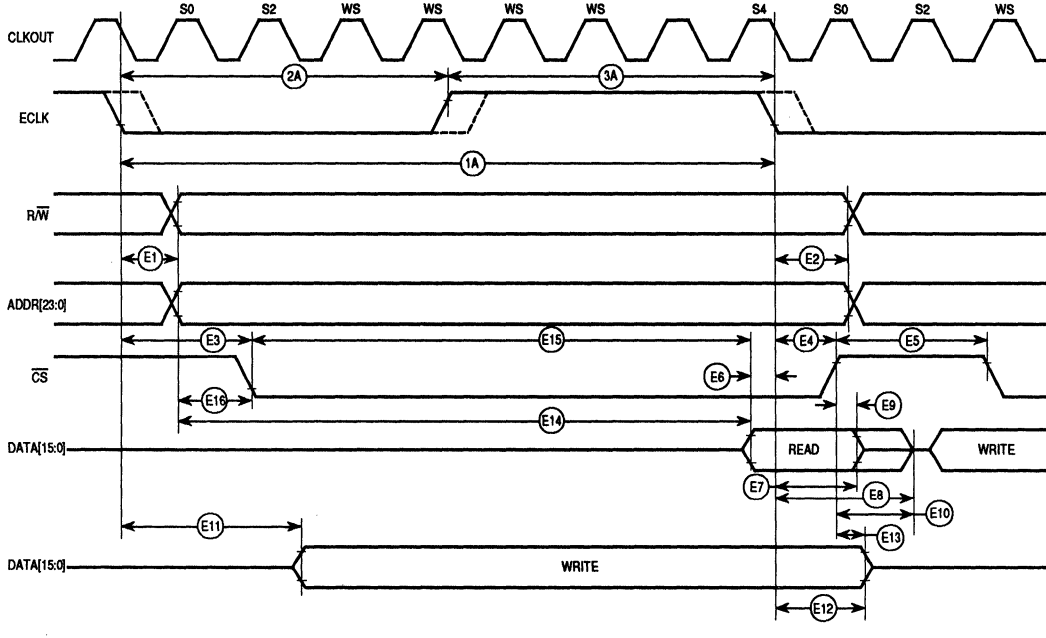
Num	Characteristic	Symbol	Min	Max	Units
6	Clock High to ADDR, FC, SIZE Valid	t_{CHAV}	0	29	ns
8	Clock High to ADDR, FC, SIZE Invalid	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Asserted	t_{CLSA}	2	25	ns
12	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated	t_{CLSN}	2	29	ns
14B	$\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Width Asserted (Fast Cycle)	t_{SWDW}	40	—	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
23	Clock High to Data Out Valid	t_{CHDO}	—	29	ns
24	Data Out Valid to Negating Edge of $\overline{AS}$, $\overline{CS}$ (Fast Write)	t_{DVASN}	15	—	ns
25	$\overline{DS}$, $\overline{CS}$ Negated to Data Out Invalid (Data Out Hold)	t_{SNDOI}	15	—	ns
46A	R/$\overline{W}$ Width Asserted (Fast Write or Read)	t_{RWAS}	90	—	ns
73	$\overline{BKPT}$ Input Setup Time	t_{BKST}	15	—	ns
74	$\overline{BKPT}$ Input Hold Time	t_{BKHT}	10	—	ns
100	CLKOUT High to Phase 1 Asserted[12]	t_{CHP1A}	3	40	ns
101	CLKOUT High to Phase 2 Asserted[12]	t_{CHP2A}	3	40	ns
102	Phase 1 Valid to $\overline{AS}$ or $\overline{DS}$ Asserted[13]	t_{P1VSA}	10	—	ns
103	Phase 2 Valid to $\overline{AS}$ or $\overline{DS}$ Negated[13]	t_{P2VSN}	10	—	ns
104	$\overline{AS}$ or $\overline{DS}$ Valid to Phase 1 Negated[13]	t_{SAP1N}	10	—	ns
105	$\overline{AS}$ or $\overline{DS}$ Negated to Phase 2 Negated[13]	t_{SNP2N}	10	—	ns

NOTES:

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

13. Eight pipeline states are multiplexed into IPIPE[1:0]. The multiplexed signals have two phases.

A



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

Figure A–12. ECLK Timing Diagram

A

Key to Figure A–12

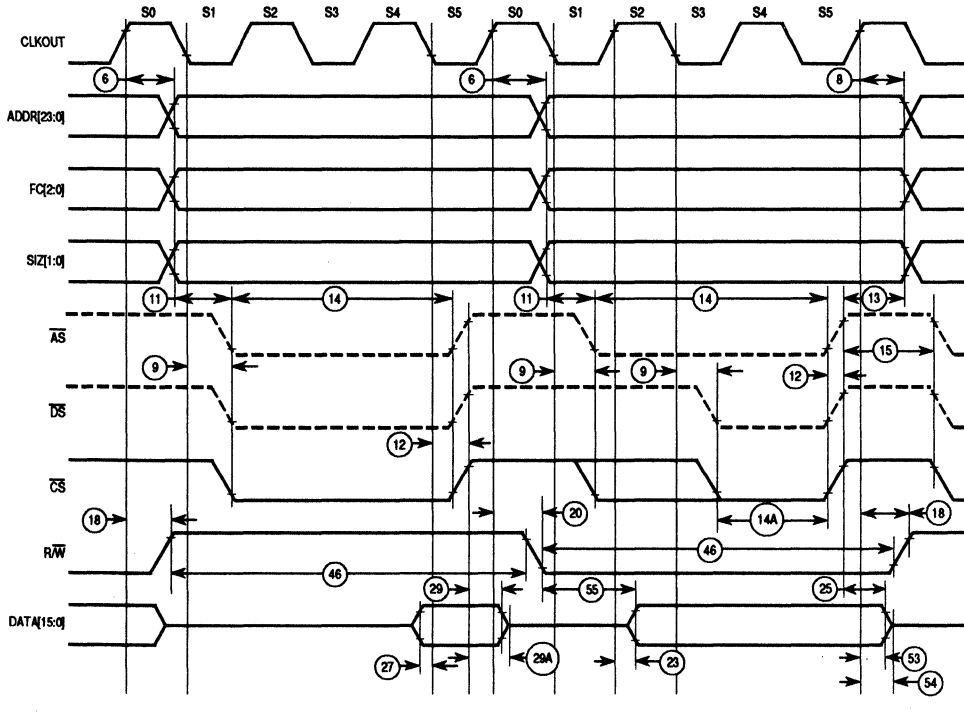
(Abstracted from Table A–7; see table for complete notes)

Num	Characteristic	Symbol	Min	Max	Units
1A	ECLK Period	t_{Ecyc}	476	—	ns
2A, 3A	ECLK Pulse Width	t_{ECW}	236	—	ns
4A, 5A	Rise and Fall Time	t_{rf}	—	8	ns
E1[2]	ECLK Low to ADDR and R/$\overline{W}$ Valid	t_{EAD}	—	60	ns
E2	ECLK Low to ADDR and R/$\overline{W}$ Hold	t_{EAH}	10	—	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}	15	—	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}	5	—	ns
E13	$\overline{CS}$ Negated to Data Hold (Write)	t_{ECHW}	0	—	ns
E14[3]	Address Access Time (Read)	t_{EACC}	386		ns
E15[4]	Chip Select Access Time (Read)	t_{EACS}	296		ns
E16	Address Setup Time	t_{EAS}	—	1/2	t_{cyc}
100	CLKOUT High to Phase 1 Asserted[12]	t_{CHP1A}	3	40	ns
101	CLKOUT High to Phase 2 Asserted[12]	t_{CHP2A}	3	40	ns
102	Phase 1 Valid to $\overline{AS}$ or $\overline{DS}$ Asserted[13]	t_{P1VSA}	10	—	ns
103	Phase 2 Valid to $\overline{AS}$ or $\overline{DS}$ Negated[13]	t_{P2VSN}	10	—	ns
104	$\overline{AS}$ or $\overline{DS}$ Valid to Phase 1 Negated[13]	t_{SAP1N}	10	—	ns
105	$\overline{AS}$ or $\overline{DS}$ Negated to Phase 2 Negated[13]	t_{SNP2N}	10	—	ns

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
2. When the previous bus cycle is not a synchronous ECLK bus 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 = $t_{Ecyc} - t_{ECSD} - t_{EDSR}$
13. Eight pipeline states are multiplexed into IPIPE[1:0]. The multiplexed signals have two phases.

A



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

Figure A–13. Chip Select Timing Diagram

Key to Figure A–13

(Abstracted from Table A–5; see table for complete notes)

Num	Characteristic	Symbol	Min	Max	Units
6	Clock High to ADDR, FC, SIZE Valid	t_{CHAV}	0	29	ns
8	Clock High to ADDR, FC, SIZE Invalid	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Asserted	t_{CLSA}	2	25	ns
11	ADDR, FC, SIZE Valid to $\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Asserted	t_{AVSA}	15	—	ns
12	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated	t_{CLSN}	2	29	ns
13	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to ADDR, FC, SIZE Invalid (Address Hold)	t_{SNAI}	15	—	ns
14	$\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Width Asserted	t_{SWA}	100	—	ns
14A	$\overline{DS}$, $\overline{CS}$ Width Asserted Write	t_{SWAW}	45	—	ns
15[5]	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Width Negated	t_{SN}	40	—	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
23	Clock High to Data Out Valid	t_{CHDO}	—	29	ns
25	$\overline{DS}$, $\overline{CS}$ Negated to Data Out Invalid (Data Out Hold)	t_{SNDOI}	15	—	ns
29[6]	$\overline{DS}$, $\overline{CS}$ Negated to Data In Invalid (Data In Hold)	t_{SNDI}	0	—	ns
29A[6,7]	$\overline{DS}$, $\overline{CS}$ Negated to Data In High Impedance	t_{SHDI}	—	55	ns
46	R/$\overline{W}$ Width Asserted (Write or Read)	t_{RWA}	150	—	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

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
5. 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 synchronized to ECLK.
6. Hold times are specified with respect to $\overline{DS}$ or $\overline{CS}$ on asynchronous reads and with respect to CLKOUT on fast reads. The user is free to use either hold time.
7. Maximum value is equal to (t_{cyc} / 2) + 25 ns.

A

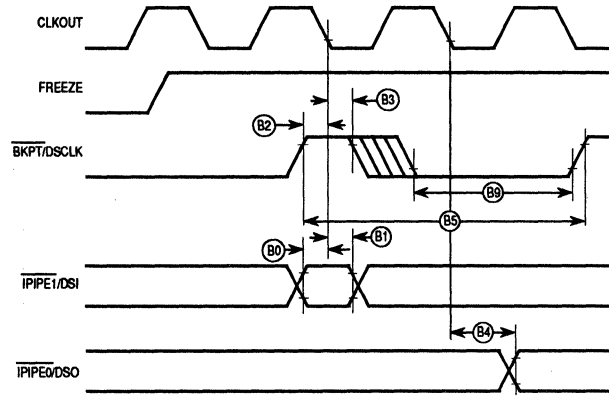


Figure A–14. Background Debugging Mode Timing Diagram — Serial Communication

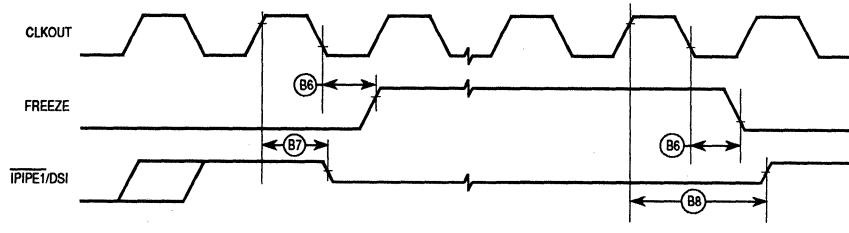


Figure A–15. Background Debugging Mode Timing Diagram — Freeze Assertion

A

Key to Figures A–14 and A–15

(Abstracted from Table A–6; see table for complete notes)

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 Low to FREEZE Asserted/Negated	t_{FRZAN}	—	50	ns
B7	CLKOUT High to IPIPE1 High Impedance	t_{IFZ}	—	TBD	ns
B8	CLKOUT High to IPIPE1 Valid	t_{IF}	—	TBD	ns
B9	DSCLK Low Time	t_{DSCLO}	1	—	t_{cyc}

NOTES:

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

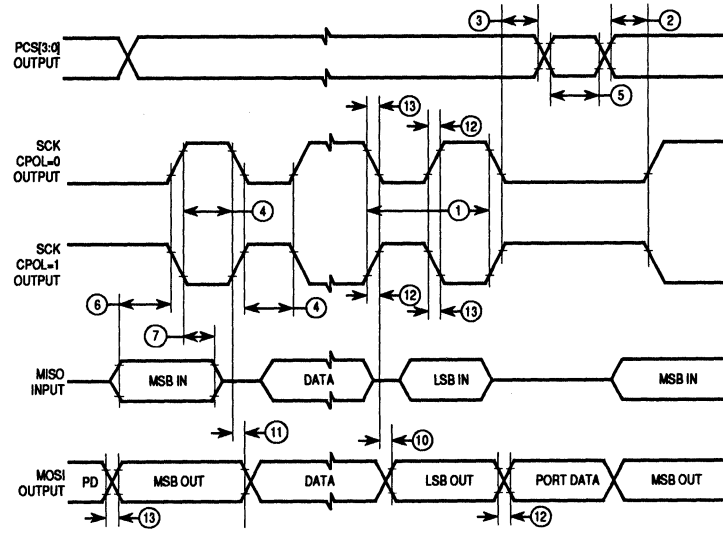


Figure A–16. QSPI Timing Master, CPHA = 0

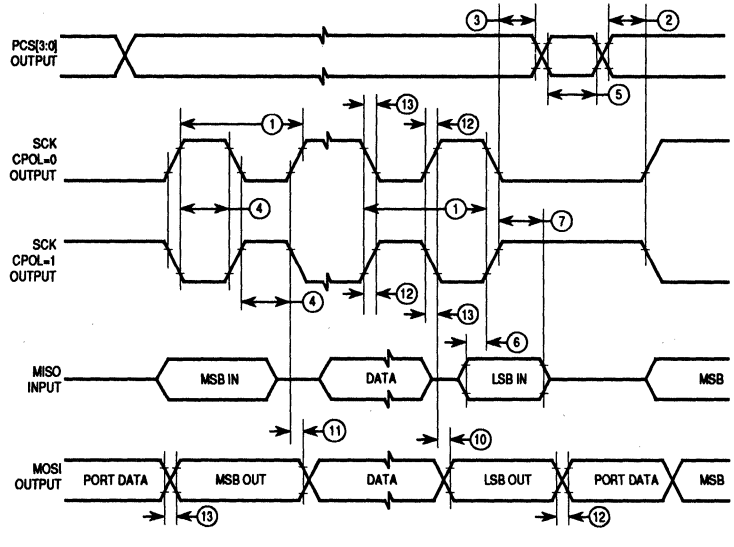


Figure A–17. QSPI Timing Master, CPHA = 1

Key to Figures A–16 and A–17

(Abstracted from Table A–8)

Num	Function	Symbol	Min	Max	Unit
1	Master Cycle Time	t_{qcyc}	4	510	t_{cyc}
2	Master Enable Lead Time	t_{lead}	2	128	t_{cyc}
3	Master Enable Lag Time	t_{lag}	—	1/2	SCK
4	Master Clock (SCK) High or Low Time	t_{sw}	2 t_{cyc}–60	255 t_{cyc}	ns
5	Master Sequential Transfer Delay	t_{td}	17	8192	t_{cyc}
6	Master Data Setup Time (Inputs)	t_{su}	30	—	ns
7	Master Data Hold Time (Inputs)	t_{hi}	0	—	ns
10	Master Data Valid (after SCK Edge)	t_v	—	50	ns
11	Master Data Hold Time (Outputs)	t_{ho}	0	—	ns
12	Rise Time Input Output	 t_{ri} t_{ro}	 — —	 2 30	 µs ns
13	Fall Time Input 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.

A

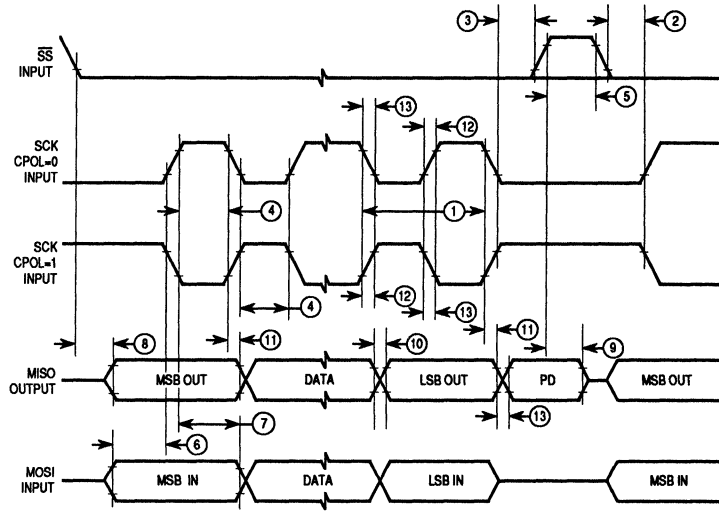


Figure A–18. QSPI Timing Slave, CPHA = 0

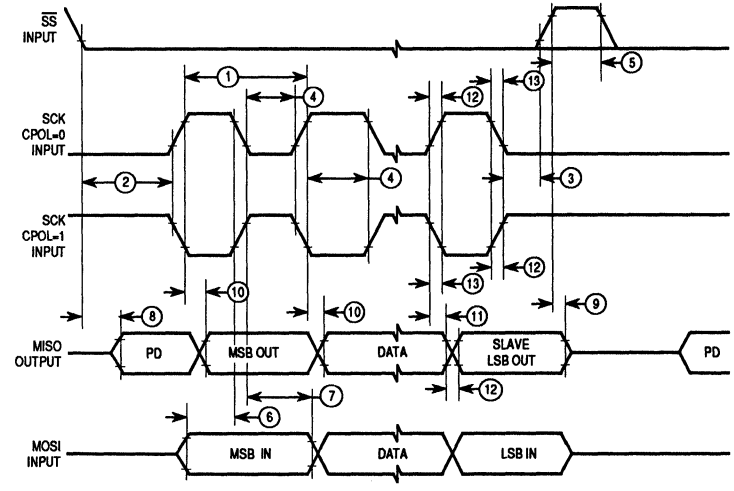


Figure A–19. QSPI Timing Slave, CPHA = 1

Key to Figures A–18 and A–19

(Abstracted from Table A–8)

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

A

A

APPENDIX B
MECHANICAL DATA AND ORDERING INFORMATION

The MC68HC16Z1 is available in a 132-pin plastic surface mount package and a 144-pin plastic surface mount package. Both packages can be ordered in molded carrier rings. This appendix provides package pin assignment drawings, dimensional drawings, and ordering information.

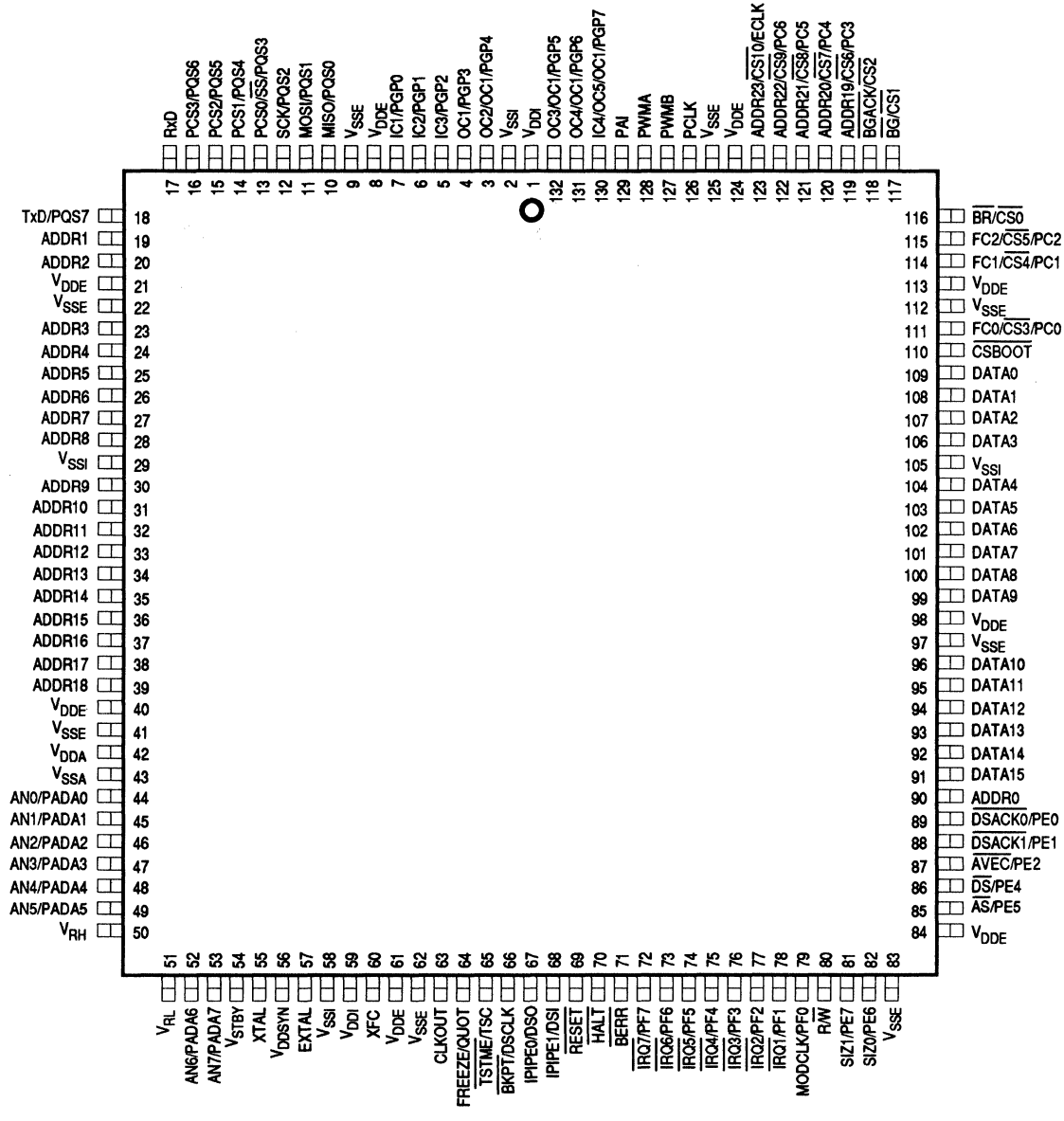
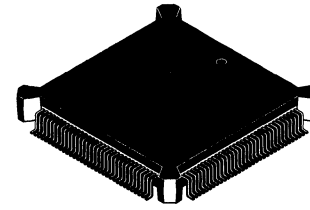
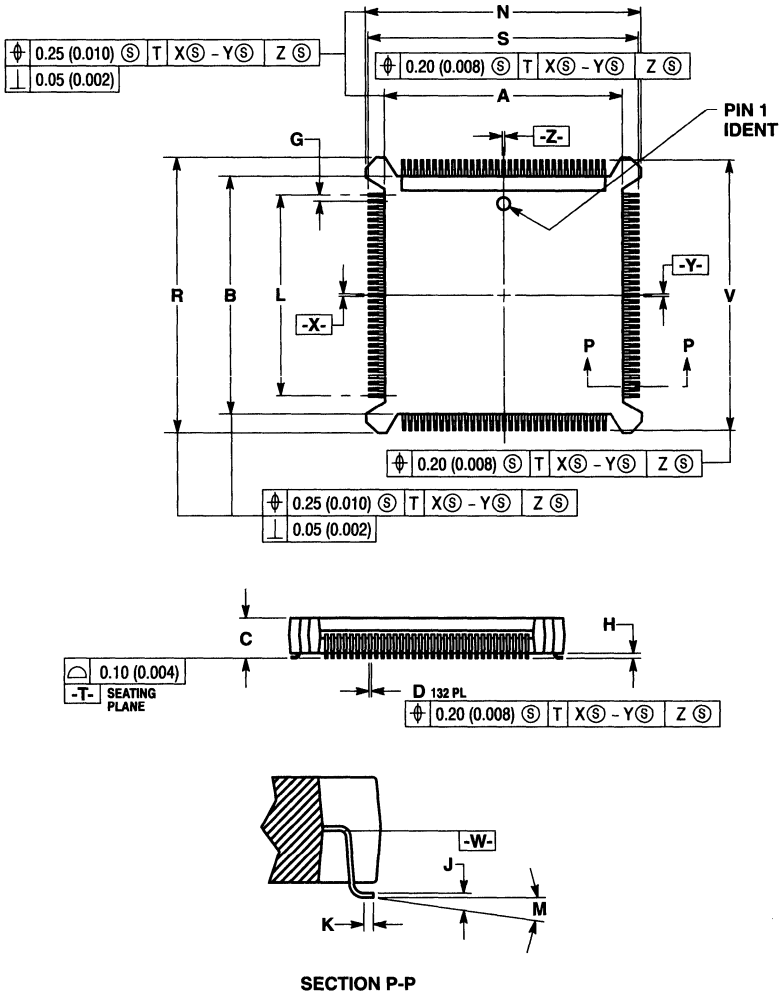


Figure B–1. 132-Pin Plastic Surface Mount Package Pin Assignments



SCALE 1:1



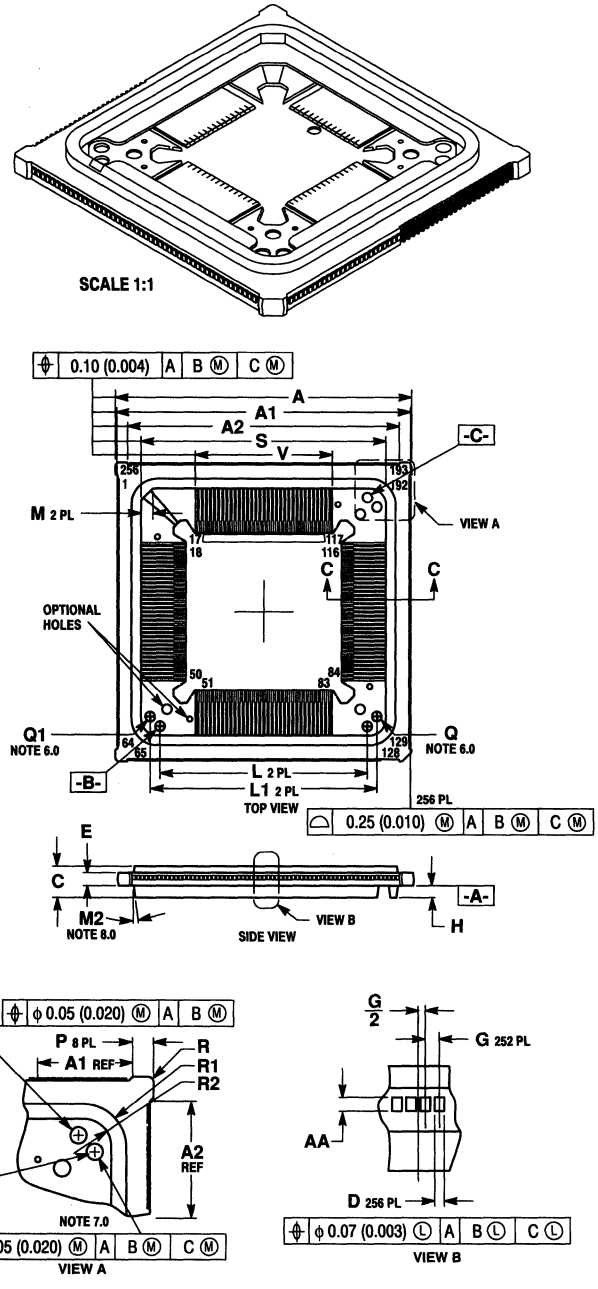
SECTION P-P

NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSIONS A, B, N, AND R DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE MOLD PROTRUSION FOR DIMENSIONS A AND B IS 0.25 (0.010), FOR DIMENSIONS N AND R IS 0.18 (0.007).
4. DATUM PLANE -W- IS LOCATED AT THE UNDERSIDE OF LEADS WHERE LEADS EXIT PACKAGE BODY.
5. DATUMS X-Y AND Z TO BE DETERMINED WHERE CENTER LEADS EXIT PACKAGE BODY AT DATUM -W-.
6. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE, DATUM -T-.
7. DIMENSIONS A, B, N, AND R TO BE DETERMINED AT DATUM PLANE -W-.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	24.06	24.20	0.947	0.953
B	24.06	24.20	0.947	0.953
C	4.07	4.57	0.160	0.180
D	0.21	0.30	0.008	0.012
G	0.64 BSC		0.025 BSC	
H	0.51	1.01	0.020	0.040
J	0.16	0.20	0.006	0.008
K	0.51	0.76	0.020	0.030
L	20.32 REF		0.800 REF	
M	0°	8°	0°	8°
N	27.88	28.01	1.097	1.103
R	27.88	28.01	1.097	1.103
S	27.31	27.55	1.075	1.085
V	27.31	27.55	1.075	1.085

CASE 831A-01

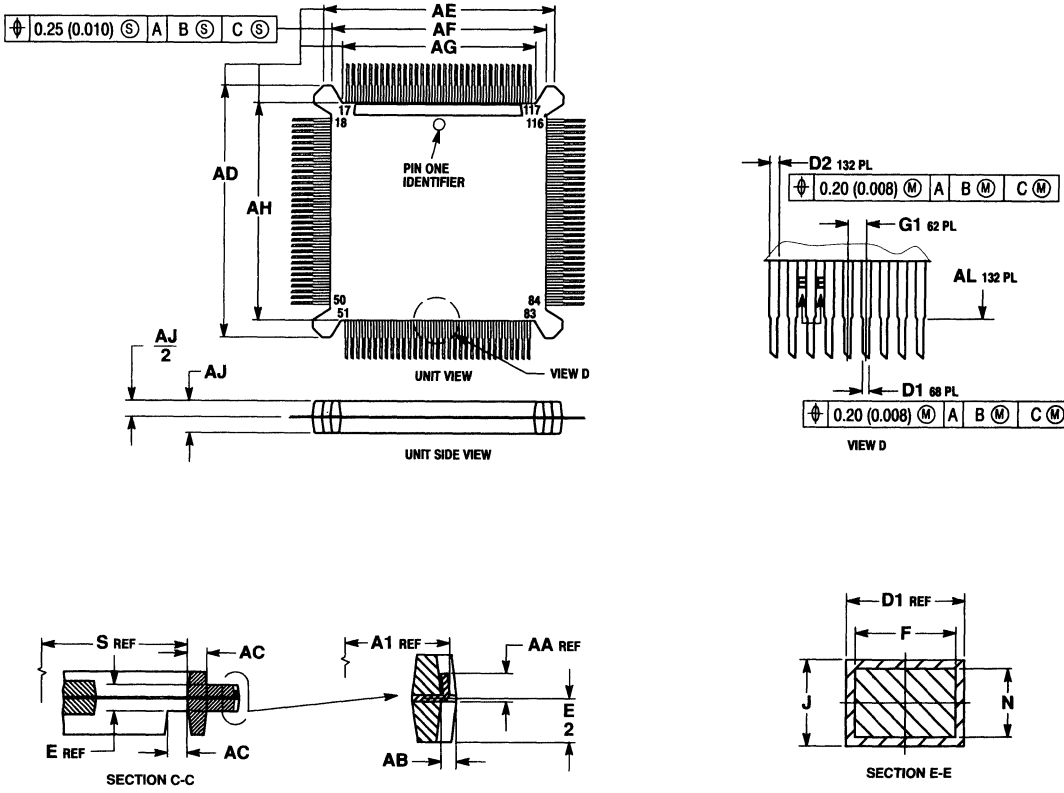
Figure B–2. 132-Pin Package Dimensions



CASE 878-01

Figure B–3. 132-Pin Molded Carrier Ring Assembly (Part 1)

B



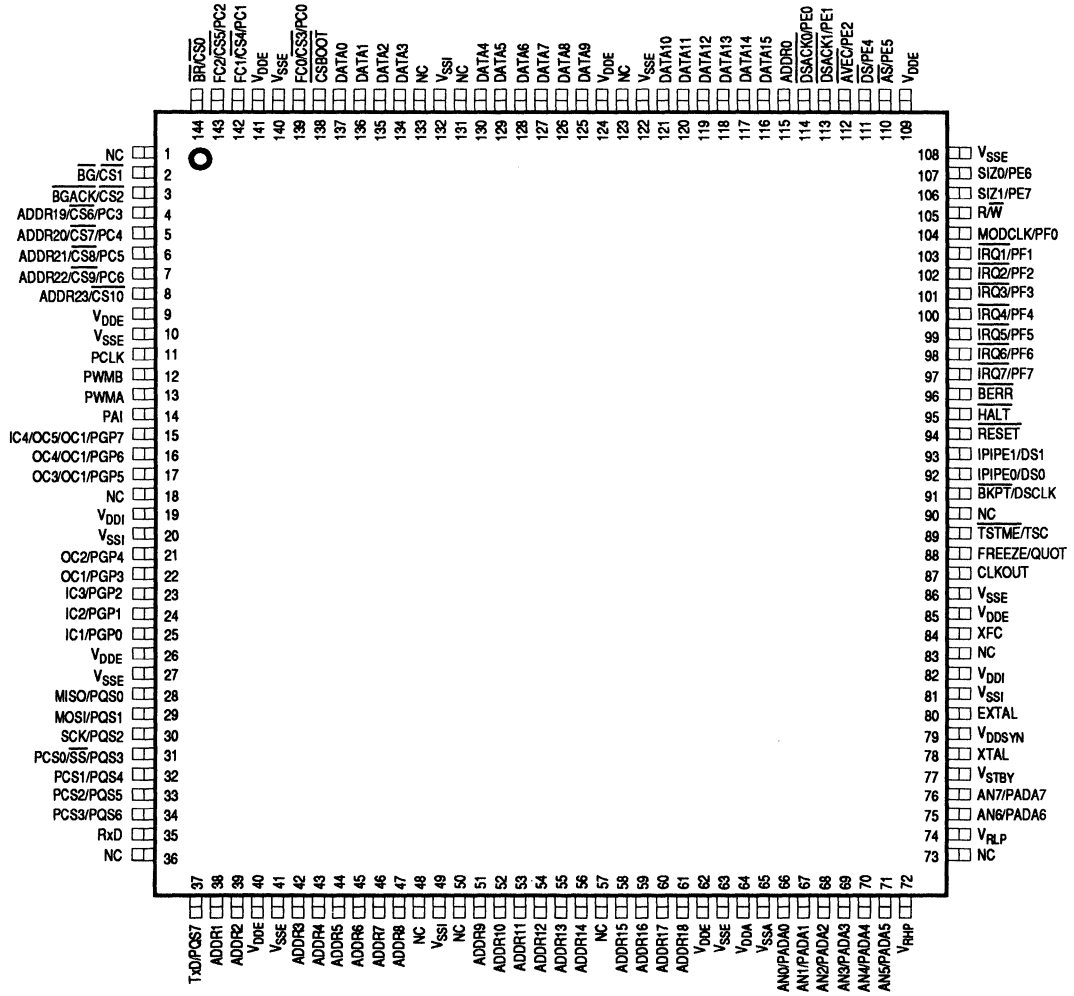
NOTES:
1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. A, AD, AE, AF AND AH DIMENSIONS DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE MOLD PROTRUSION IS 0.20 (0.008) PER SIDE. FOR AE DIMENSION IS 0.18 (0.007).
4. A, S1 AND AH DIMENSIONS INCLUDE MOLD MISMATCH, AND ARE MEASURED AT THE PARTING LINE.
5. UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE SYMMETRICAL ABOUT CENTERLINES.
6. B AND C DATUM HOLES ARE TO BE USED FOR TRIM, FORM AND EXCISE OF THE MOLDED PACKAGE ONLY. HOLES Q1 AND Q2 ARE TO BE USED FOR ELECTRICAL TESTING ONLY.
7. NON-DATUM HOLES ONLY.
8. APPLIES TO RING AND PACKAGE FEATURES.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	45.87	46.13	1.806	1.816
A1	45.70 BSC		1.799 BSC	
A2	41.37	41.63	1.629	1.639
C	4.70	4.90	0.185	0.193
D	0.40	0.50	0.016	0.020
D1	0.21	0.30	0.008	0.012
D2	0.31	0.40	0.012	0.016
E	1.90	2.10	0.075	0.083
F	0.19	0.27	0.007	0.011
G	0.65 BSC		0.026 BSC	
G1	0.635 BSC		0.025 BSC	
H	1.70	1.90	0.067	0.075
J	0.16	0.20	0.006	0.008
L	32.20 BSC		1.268 BSC	
L1	35.20 BSC		1.386 BSC	
M	1.30	2.30	0.051	0.091
M2	6°	8°	6°	8°
N	0.145	0.16	0.0057	0.0063
P	1.77	2.03	0.070	0.080

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
R	0.40	0.60	0.016	0.024
R1	3.50	4.50	0.138	0.177
R2	2.00	3.00	0.079	0.118
S	37.87	38.13	1.491	1.501
V	21.21	21.31	0.835	0.839
W	1.45	1.55	0.057	0.061
AA	0.45	0.85	0.018	0.033
AB	0.30	0.60	0.012	0.024
AC	1.37	1.63	0.054	0.064
AD	27.88	28.01	1.098	1.103
AE	25.79	25.93	1.015	1.021
AF	23.91	24.05	0.941	0.947
AG	21.52	21.66	0.847	0.853
AH	24.06	24.20	0.947	0.953
AJ	3.46	3.66	0.136	0.144
AL	14.00	14.10	0.551	0.555

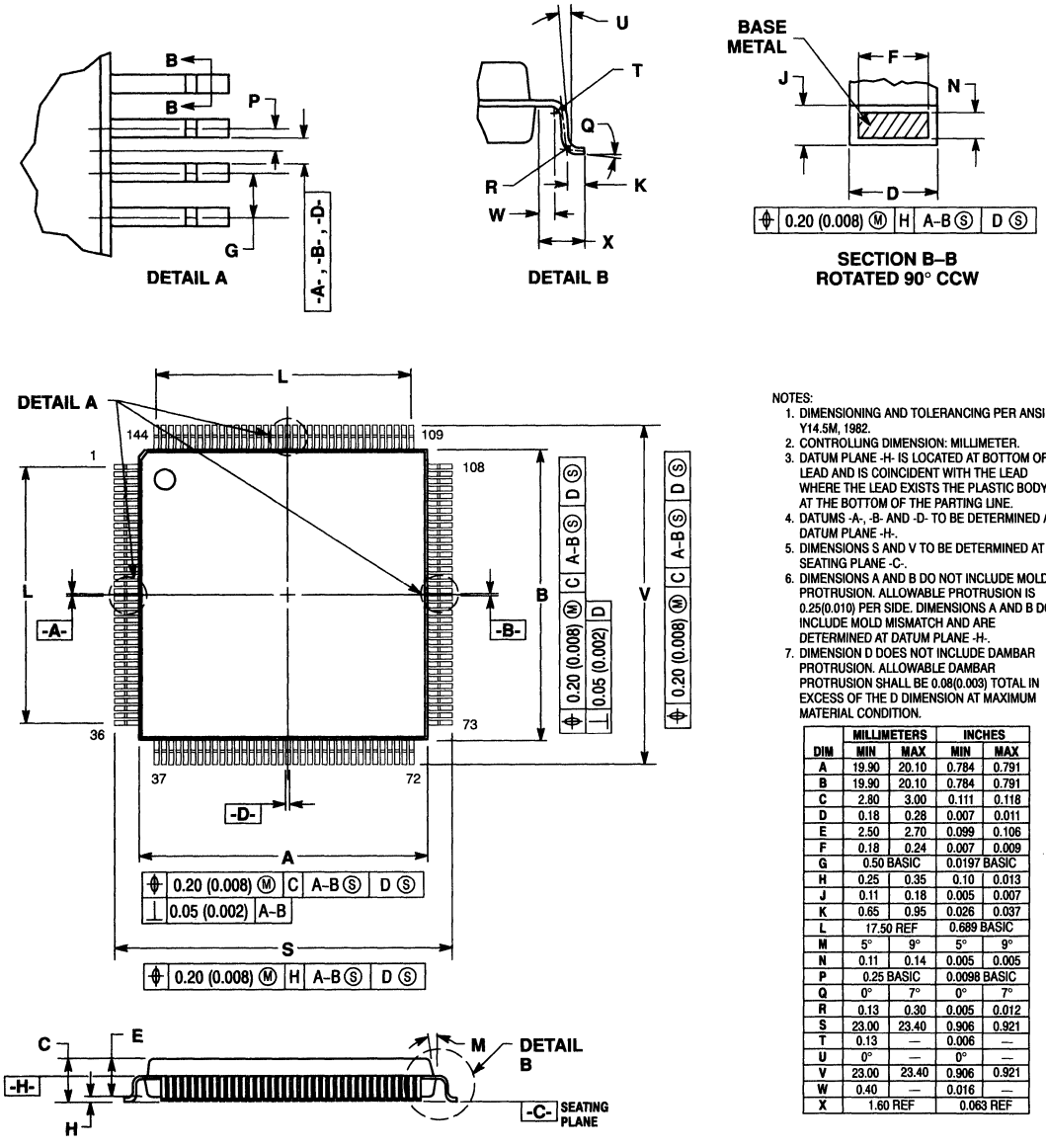
B

Figure B–3. 132-Pin Molded Carrier Ring Assembly (Part 2)



B

Figure B–4. 144-Pin Plastic Surface Mount Package Pin Assignments



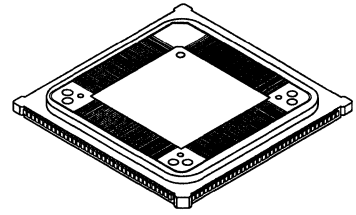
NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. DATUM PLANE -H- IS LOCATED AT BOTTOM OF LEAD AND IS COINCIDENT WITH THE LEAD WHERE THE LEAD EXISTS THE PLASTIC BODY AT THE BOTTOM OF THE PARTING LINE.
4. DATUMS -A-, -B- AND -D- TO BE DETERMINED AT DATUM PLANE -H-.
5. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE -C-.
6. DIMENSIONS A AND B DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25(0.010) PER SIDE. DIMENSIONS A AND B DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE -H-.
7. DIMENSION D DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.08(0.003) TOTAL IN EXCESS OF THE D DIMENSION AT MAXIMUM MATERIAL CONDITION.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	19.90	20.10	0.784	0.791
B	19.90	20.10	0.784	0.791
C	2.80	3.00	0.111	0.118
D	0.18	0.28	0.007	0.011
E	2.50	2.70	0.099	0.106
F	0.18	0.24	0.007	0.009
G	0.50 BASIC		0.0197 BASIC	
H	0.25	0.35	0.10	0.013
J	0.11	0.18	0.005	0.007
K	0.65	0.95	0.026	0.037
L	17.50 REF		0.689 BASIC	
M	5°	9°	5°	9°
N	0.11	0.14	0.005	0.005
P	0.25 BASIC		0.0098 BASIC	
Q	0°	7°	0°	7°
R	0.13	0.30	0.005	0.012
S	23.00	23.40	0.906	0.921
T	0.13	—	0.006	—
U	0°	—	0°	—
V	23.00	23.40	0.906	0.921
W	0.40	—	0.016	—
X	1.60 REF		0.063 REF	

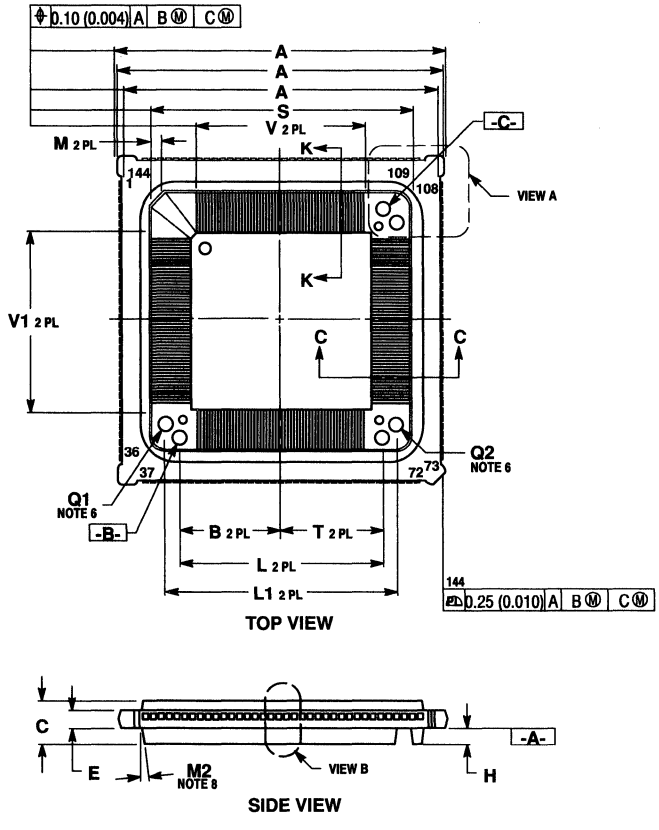
CASE 863C-01

Figure B–5. 144-Pin Package Dimensions

B



SCALE 1:1



NOTES:
1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. A, DIMENSION DOES NOT INCLUDE MOLD PROTRUSION. ALLOWABLE MOLD PROTRUSION IS 0.20 (0.008) PER SIDE
4. A AND S DIMENSIONS INCLUDE MOLD MISMATCH, AND ARE MEASURED AT THE PARTING LINE.
5. UNLESS OTHEWISE SPECIFIED DIMENSIONS ARE SYMMETRICAL ABOUT CENTERLINES.
6. B AND C DATUM HOLES ARE TO BE USED FOR TRIM, FORM AND EXCISE OF THE MOLDED PACKAGE ONLY. HOLES Q1 AND Q2 ARE TO BE USED FOR ELECTRICAL TESTING ONLY.
7. NON-DATUM HOLES ONLY.
8. APPLIES TO RING AND PACKAGE FEATURES.
9. DIMENSION D1 DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.08 (0.003) TOTAL IN EXCESS OF THE DIMENSION AT MAXIMUM MATERIAL CONDITION.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	35.870	36.130	1.412	1.422
A1	35.700 BSC		1.406 BSC	
A2	35.180	35.430	1.385	1.395
B	11.000 BSC		0.433 BSC	
C	4.700	4.900	0.185	0.193
D	0.410	0.500	0.016	0.020
D1	0.150	0.304	0.006	0.012
D2	0.400 MAX		0.016 MAX	
E	1.910	2.109	0.075	0.083
F	0.150	0.260	0.006	0.010
G	0.650 BSC		0.026 BSC	
G1	0.500 BSC		0.020 BSC	
H	1.700	1.90	0.067	0.075
J	0.130	0.23	0.006	0.009
L	22.000 BSC		0.866 BSC	
L1	25.200 BSC		0.992 BSC	
M	1.600	2.000	0.063	0.078
M2	8° MAX		8° MAX	
N	0.130	0.170	0.005	0.007
P	1.770	2.030	0.070	0.080
R	0.400	0.600	0.016	0.024
R1	3.500	4.500	0.138	0.177
R2	2.000	3.000	0.079	0.118
S	27.870	28.130	1.097	1.107
T	11.000 BSC		0.433 BSC	
V	14.550	14.750	0.573	0.580
V1	14.550	14.750	0.573	0.580
W	1.450	1.550	0.057	0.061
AA	0.450	0.850	0.018	0.033
AB	0.300	0.600	0.012	0.024
AC	1.370	1.630	0.054	0.064
AD	1.370	1.630	0.054	0.064
AH	19.900	20.100	0.7835	0.7913
AH1	19.900	20.100	0.7835	0.7913
AJ	2.530	2.670	0.0996	0.1051

CASE 922-01

Figure B–6. 144-Pin Molded Carrier Ring Assembly (Part 1)

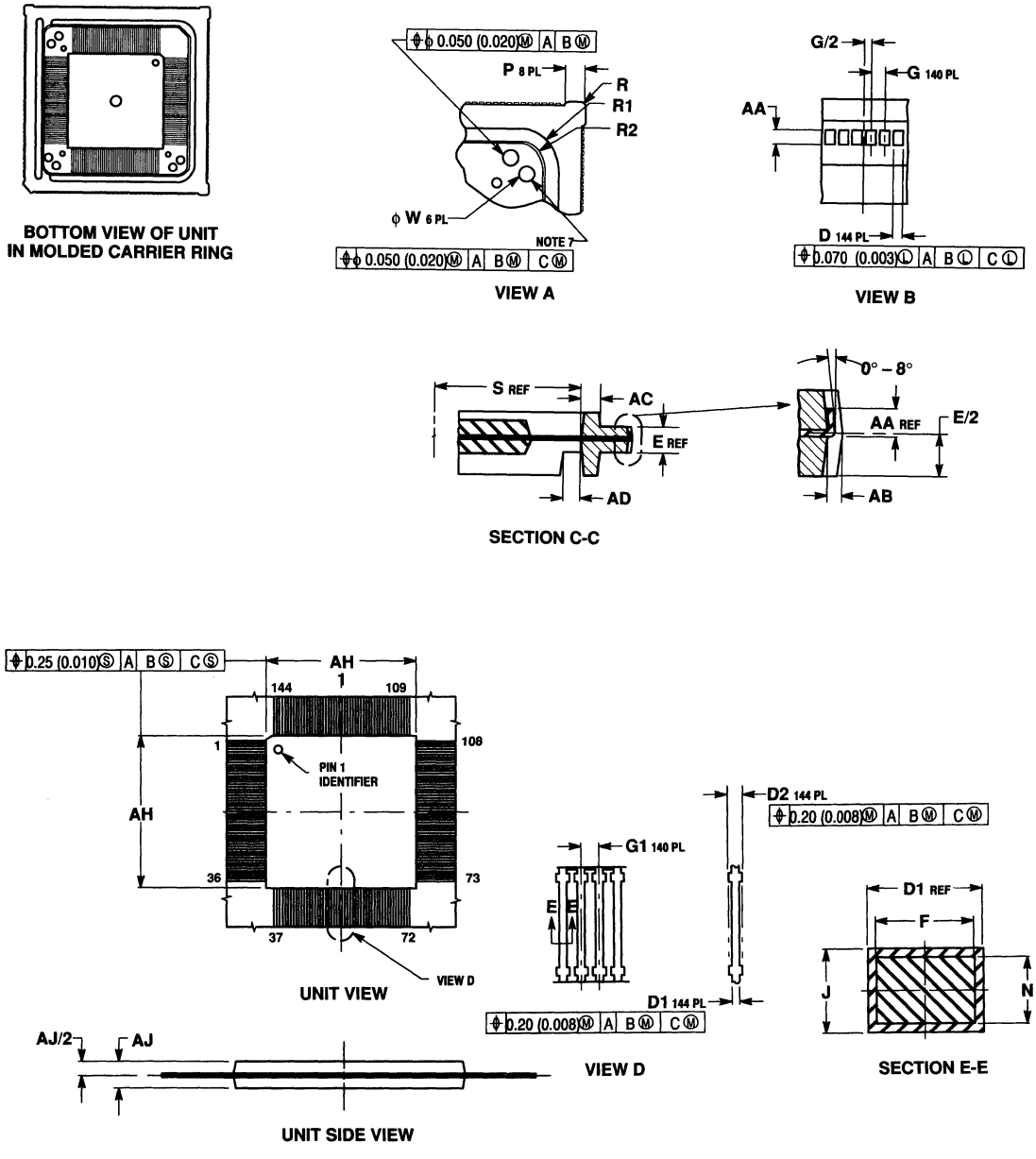


Figure B–6. 144-Pin Molded Carrier Ring Assembly (Part 2)

Table B–1. MC68HC16Z1 Ordering Information

Device Package	Temperature Range (°C)	Reference Frequency	Shipping Method	Order Number
132-PIN PLASTIC SURFACE MOUNT	–40 to 85	16.78 MHz	36 PER TRAY	XC16Z1CFC16
			2 PER TRAY	SPAKXC16Z1CFC16
		20 MHz	36 PER TRAY	XC16Z1CFC20
			2 PER TRAY	SPAKXC16Z1CFC20
		25 MHz	36 PER TRAY	XC16Z1CFC25
			2 PER TRAY	SPAKXC16Z1CFC25
	–40 to 105	16.78 MHz	36 PER TRAY	XC16Z1VFC16
			2 PER TRAY	SPAKXC16Z1VFC16
		20 MHz	36 PER TRAY	XC16Z1VFC20
			2 PER TRAY	SPAKXC16Z1VFC20
		25 MHz	36 PER TRAY	XC16Z1VFC25
			2 PER TRAY	SPAKXC16Z1VFC25
	–40 to 125	16.78 MHz	36 PER TRAY	XC16Z1MFC16
			2 PER TRAY	SPAKXC16Z1MFC16
		20 MHz	36 PER TRAY	XC16Z1MFC20
			2 PER TRAY	SPAKXC16Z1MFC20
		25 MHz	36 PER TRAY	XC16Z1MFC25
			2 PER TRAY	SPAKXC16Z1MFC25
132-PIN MOLDED CARRIER RING	–40 to 85	16.78 MHz	10 PER TUBE	XC16Z1CFD16
		20 MHz	10 PER TUBE	XC16Z1CFD20
		25 MHz	10 PER TUBE	XC16Z1CFD25
	–40 to 105	16.78 MHz	10 PER TUBE	XC16Z1VFD16
		20 MHz	10 PER TUBE	XC16Z1VFD20
		25 MHz	10 PER TUBE	XC16Z1VFD25
	–40 to 125	16.78 MHz	10 PER TUBE	XC16Z1MFD16
		20 MHz	10 PER TUBE	XC16Z1MFD20
		25 MHz	10 PER TUBE	XC16Z1MFD25
144-PIN PLASTIC SURFACE MOUNT	–40 to 85	16.78 MHz	44 PER TRAY	XC16Z1CFV16
			2 PER TRAY	SPAKXC16Z1CFV16
		20 MHz	44 PER TRAY	XC16Z1CFV20
			2 PER TRAY	SPAKXC16Z1CFV20
		25 MHz	44 PER TRAY	XC16Z1CFV25
			2 PER TRAY	SPAKXC16Z1CFV25
	–40 to 105	16.78 MHz	44 PER TRAY	XC16Z1VFV16
			2 PER TRAY	SPAKXC16Z1VFV16
		20 MHz	44 PER TRAY	XC16Z1VFV20
			2 PER TRAY	SPAKXC16Z1VFV20
		25 MHz	44 PER TRAY	XC16Z1VFV25
			2 PER TRAY	SPAKXC16Z1VFV25
	–40 to 125	16.78 MHz	44 PER TRAY	XC16Z1MFV16
			2 PER TRAY	SPAKXC16Z1MFV16
		20 MHz	44 PER TRAY	XC16Z1MFV20
			2 PER TRAY	SPAKXC16Z1MFV20
		25 MHz	44 PER TRAY	XC16Z1MFV25
			2 PER TRAY	SPAKXC16Z1MFV25
144-PIN MOLDED CARRIER RING	–40 to 85	16.78 MHz	13 PER TUBE	XC16Z1CFM16
		20 MHz	13 PER TUBE	XC16Z1CFM20
		25 MHz	13 PER TUBE	XC16Z1CFM25
	–40 to 105	16.78 MHz	13 PER TUBE	XC16Z1VFM16
		20 MHz	13 PER TUBE	XC16Z1VFM20
		25 MHz	13 PER TUBE	XC16Z1VFM25
	–40 to 125	16.78 MHz	13 PER TUBE	XC16Z1MFM16
		20 MHz	13 PER TUBE	XC16Z1MFM20
		25 MHz	13 PER TUBE	XC16Z1MFM25

APPENDIX C
DEVELOPMENT SUPPORT

This section is a brief reference to Motorola development tools for the MC68HC16Z1 microcontroller. Information provided is complete at the time of publication, but new systems and software are continually being developed. In addition, there are a growing number of third-party tools available. The Motorola *MCU Tool Box* (MCUTLBX/D Rev. B) provides an up-to-date list of development tools. Contact your Motorola representative for further information.

Table C–1. MC68HC16Z1 Development Tools

Microcontroller Part Number	Evaluation Board	Evaluation Modules*	Evaluation Systems*/Kits	Programmer Boards
MC68HC16Z1	M68HC16Z1EVB	—	—	—

*EVS and EVM include an Integrated Development Environment (IDE) which contains an editor, assembler, hardware debugger and simulator.

C.1 M68HC16EVB Evaluation Board

The evaluation board is a low-cost tool for debugging and evaluating MC68HC16Z1-based target systems. Features of the M68HC16EVB include:

- Operation in either debugging or evaluation mode.
- Operation from a personal computer platform without an on-board monitor using CPU16 background debugging mode.
- An integrated assembly/editing/emulation environment.
- As many as seven software breakpoints.
- Memory map of target system.
- On-board RS-232C port.
- Logic analyzer pod connectors.

Refer to the *M68HC16EVB Evaluation Board User's Manual*, (M68HC16Z1EVB/D) for more information.

C

C

APPENDIX D
REGISTER SUMMARY

This appendix contains address maps, register diagrams, and bit/field definitions for the MC68HC16Z1. This appendix is intended to be a ready reference. In-depth 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. MC68HC16Z1 Module Address Map

Module	Size (Bytes)	Address Bus Decoding 23 12 11 0						Base Address
ADC	64	M111	1111	1111	0111	00XX	XXXX	$YFF700
GPT	64	M111	1111	1111	1001	00XX	XXXX	$YFF900
SIM	128	M111	1111	1111	1010	0XXX	XXXX	$YFFA00
SRAM	8	M111	1111	1111	1011	0000	0XXX	$YFFB00
QSM	512	M111	1111	1111	110X	XXXX	XXXX	$YFFC00

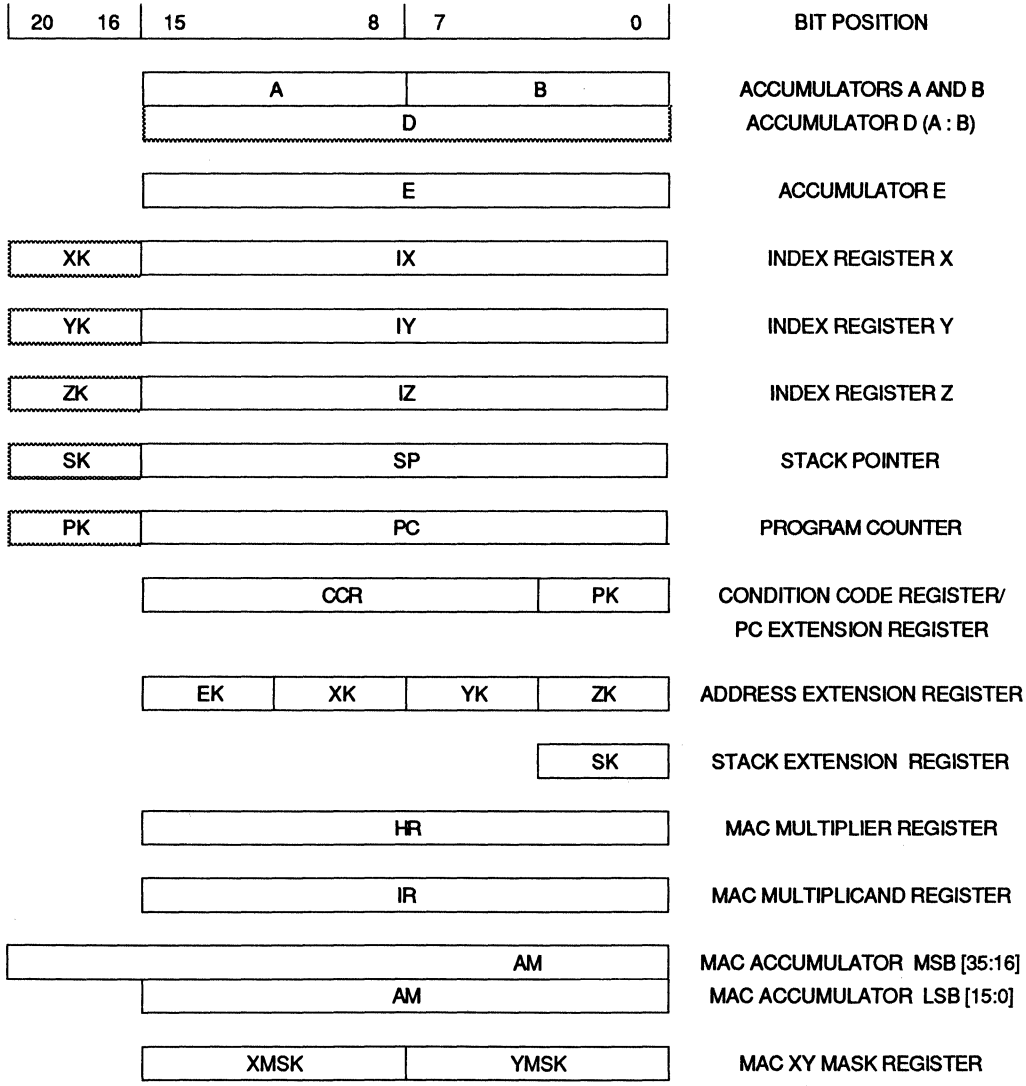
Control registers for all the modules in the microcontroller are mapped into a 4 Kbyte block. The state of the module mapping (MM) bit in the SIM module configuration register (SIMCR) determines where the control registers block is located in the system memory map. When MM = 0, register addresses range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

In the MC68HC16Z1, ADDR[23:20] follow the logic state of ADDR19 unless externally driven. MM corresponds to IMB ADDR23 — if it is cleared, the SIM maps IMB modules into address space $7FF000–$7FFFFF, which is inaccessible to the CPU16. Modules remain inaccessible until reset occurs. The reset state of MM is one, but the bit is one-time writable — initialization software should make certain it remains set.

D.1 Central Processing Unit

CPU16 registers are not part of the module address map. The following diagram is a functional representation of CPU resources.

D.1.1 CPU16 Register Model



D.1.2 CCR — Condition Code Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S	MV	H	EV	N	Z	V	C	IP			SM	PK			

The CCR contains processor status flags, the interrupt priority field, and the program counter address extension field. The CPU16 has a special set of instructions that manipulate the CCR.

S — STOP Enable
0 = Stop clock when LPSTOP instruction is executed.
1 = Perform NOP when LPSTOP instruction is executed.

MV — Accumulator M overflow flag
Set when overflow into AM35 has occurred.

H — Half Carry Flag
Set when a carry from A3 or B3 occurs during BCD addition.

EV — Extension Bit Overflow Flag
Set when an overflow into AM31 has occurred.

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 twos complement overflow occurs as the result of an operation.

C — Carry Flag
Set when carry or borrow occurs during arithmetic operation. Also used during shift and rotate to facilitate multiple word operations.

IP[2:0] — Interrupt Priority Field
The priority value in this field (0 to 7) is used to mask interrupts.

SM — Saturate Mode Bit
When SM is set, if either EV or MV is set, data read from AM using TMER or TMET will be given maximum positive or negative value, depending on the state of the AM sign bit before overflow.

PK[3:0] — Program Counter Address Extension Field
This field is concatenated with the program counter to form a 20-bit address.

Table D–2. ADC Module Address Map

Address	15 8	7 0
$YFF700	MODULE CONFIGURATION (ADCMCR)	
$YFF702	FACTORY TEST (ADTEST)	
$YFF704	(RESERVED)	
$YFF706	PORT ADA DATA (PORTADA)	
$YFF708	(RESERVED)	
$YFF70A	ADC CONTROL 0 (ADCTL0)	
$YFF70C	ADC CONTROL 1 (ADCTL1)	
$YFF70E	ADC STATUS (ADSTAT)	
$YFF710	RIGHT-JUSTIFIED UNSIGNED RESULT 0 (RJURR0)	
$YFF712	RIGHT-JUSTIFIED UNSIGNED RESULT 1 (RJURR1)	
$YFF714	RIGHT-JUSTIFIED UNSIGNED RESULT 2 (RJURR2)	
$YFF716	RIGHT-JUSTIFIED UNSIGNED RESULT 3 (RJURR3)	
$YFF718	RIGHT-JUSTIFIED UNSIGNED RESULT 4 (RJURR4)	
$YFF71A	RIGHT-JUSTIFIED UNSIGNED RESULT 5 (RJURR5)	
$YFF71C	RIGHT-JUSTIFIED UNSIGNED RESULT 6 (RJURR6)	
$YFF71E	RIGHT-JUSTIFIED UNSIGNED RESULT 7 (RJURR7)	
$YFF720	LEFT-JUSTIFIED SIGNED RESULT 0 (LJSRR0)	
$YFF722	LEFT-JUSTIFIED SIGNED RESULT 1 (LJSRR1)	
$YFF724	LEFT-JUSTIFIED SIGNED RESULT 2 (LJSRR2)	
$YFF726	LEFT-JUSTIFIED SIGNED RESULT 3 (LJSRR3)	
$YFF728	LEFT-JUSTIFIED SIGNED RESULT 4 (LJSRR4)	
$YFF72A	LEFT-JUSTIFIED SIGNED RESULT 5 (LJSRR5)	
$YFF72C	LEFT-JUSTIFIED SIGNED RESULT 6 (LJSRR6)	
$YFF72E	LEFT-JUSTIFIED SIGNED RESULT 7 (LJSRR7)	
$YFF730	LEFT-JUSTIFIED UNSIGNED RESULT 0 (LJURR0)	
$YFF732	LEFT-JUSTIFIED UNSIGNED RESULT 1 (LJURR1)	
$YFF734	LEFT-JUSTIFIED UNSIGNED RESULT 2 (LJURR2)	
$YFF736	LEFT-JUSTIFIED UNSIGNED RESULT 3 (LJURR3)	
$YFF738	LEFT-JUSTIFIED UNSIGNED RESULT 4 (LJURR4)	
$YFF73A	LEFT-JUSTIFIED UNSIGNED RESULT 5 (LJURR5)	
$YFF73C	LEFT-JUSTIFIED UNSIGNED RESULT 6 (LJURR6)	
$YFF73E	LEFT-JUSTIFIED UNSIGNED RESULT 7 (LJURR7)	

D.2.1 ADCMCR — ADC Module Configuration Register $YFF700

15	14	13	12	8	7	6	0
STOP	FRZ		NOT USED		SUPV	NOT USED	
RESET:							
1	0	0			0		

ADCMCR controls ADC operation during low-power stop and freeze modes.

STOP — STOP Mode

0 = Normal operation

1 = Low-power operation

STOP places the ADC in low-power state. Setting STOP aborts any conversion in progress. STOP is set during reset and must be cleared before ADC operation can begin — because analog circuitry bias current has been turned off, there must be a period of recovery after STOP is cleared before conversion begins.

FRZ[1:0] — Freeze

The FRZ field determines ADC response to assertion of the FREEZE signal.

Freeze Encoding

FRZ[1:0]	Response
00	Ignore IFREEZE
01	Reserved
10	Finish conversion, then freeze
11	Freeze immediately

SUPV — Supervisor/Unrestricted

Because the CPU16 in the MC68HC16Z1 operates only in supervisor mode, this bit has no effect.

D.2.2 ADTEST — ADC Test Register $YFF702

ADTEST is used with the SIM test register for factory test of the ADC.

D.2.3 PORTADA — Port ADA Data Register $YFF706

15							8	7	0
NOT USED								PADA	
RESET:									
0	0	0	0	0	0	0	0	INPUT DATA	

A read of PADA[7:0] returns the logic levels of port ADA pins. If an input is outside specified logic levels, an indeterminate value is read. Use as a digital input does not preclude use as an analog input.

D

D.2.4 ADCTL0 — A/D Control Register 0 $YFF70A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								RES10	STS		PRS				
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

ADCTL0 is used to select resolution, sample time, and clock/prescaler value. Writing ADCTL0 aborts any conversion in progress — ADC activity halts until ADCTL1 is written.

RES10 — 10-Bit Resolution

0 = 8-bit conversion

1 = 10-bit conversion

STS[1:0] — Sample Time Selection

The STS field selects one of four sample times.

Sample Time Selection

STS[1:0]	Sample Time
00	4 A/D Clock Periods
01	8 A/D Clock Periods
10	16 A/D Clock Periods
11	32 A/D Clock Periods

PRS[4:0] — Prescaler Rate Selection

ADC clock is generated from system clock using a modulus counter and a divide-by-two circuit. PRS contains the counter modulus.

ADC Clock Selection

PRS[4:0]	ADCCLK
%00000	RESERVED
%00001	Sys Clk/4
%00010	Sys Clk/6
...	...
%11101	Sys Clk/60
%11110	Sys Clk/62
%11111	Sys Clk/64

D

D.2.5 ADCTL1 — A/D Control Register 1 $YFF70C

15								7	6	5	4	3	2	1	0
NOT USED									SCAN	MULT	S8CM	CD	CC	CB	CA
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADCTL1 selects conversion mode and the analog channel or channels. Writing ADCTL1 aborts any conversion in progress and initiates a new conversion sequence.

SCAN — Scan Mode Selection
- 0 = Single conversion sequence
- 1 = Continuous conversion

MULT — Multichannel Conversion
- 0 = Conversion sequence(s) run on a single channel selected by [CD:CA].
- 1 = Sequential conversion of four or eight channels selected by [CD:CA].

S8CM — Select Eight-Conversion Sequence Mode
- 0 = Four-conversion sequence
- 1 = Eight-conversion sequence

ADC Conversion Modes

SCAN	MULT	S8CM	MODE
0	0	0	SINGLE 4-CONVERSION SINGLE-CHANNEL SEQUENCE
0	0	1	SINGLE 8-CONVERSION SINGLE-CHANNEL SEQUENCE
0	1	0	SINGLE 4-CONVERSION MULTICHANNEL SEQUENCE
0	1	1	SINGLE 8-CONVERSION MULTICHANNEL SEQUENCE
1	0	0	MULTIPLE 4-CONVERSION SINGLE-CHANNEL SEQUENCES
1	0	1	MULTIPLE 8-CONVERSION SINGLE-CHANNEL SEQUENCES
1	1	0	MULTIPLE 4-CONVERSION MULTICHANNEL SEQUENCES
1	1	1	MULTIPLE 8-CONVERSION MULTICHANNEL SEQUENCES

[CD:CA] — Channel Selection

Bits in this field select input channel or channels for analog-to-digital conversion.

Conversion mode determines which channel or channels are selected for conversion and which result registers are used to store conversion results. The following tables summarize the effects of ADCTL1 bits and fields.

Single-Channel Conversions

S8CM	CD	CC	CB	CA	Input	Result Register
0	0	0	0	0	AN0	RSLT[0:3]
0	0	0	0	1	AN1	RSLT[0:3]
0	0	0	1	0	AN2	RSLT[0:3]
0	0	0	1	1	AN3	RSLT[0:3]
0	0	1	0	0	AN4	RSLT[0:3]
0	0	1	0	1	AN5	RSLT[0:3]
0	0	1	1	0	AN6	RSLT[0:3]
0	0	1	1	1	AN7	RSLT[0:3]
0	1	0	0	0	RESERVED	RSLT[0:3]
0	1	0	0	1	RESERVED	RSLT[0:3]
0	1	0	1	0	RESERVED	RSLT[0:3]
0	1	0	1	1	RESERVED	RSLT[0:3]
0	1	1	0	0	V_{RH}	RSLT[0:3]
0	1	1	0	1	V_{RL}	RSLT[0:3]
0	1	1	1	0	$(V_{RH} - V_{RL}) / 2$	RSLT[0:3]
0	1	1	1	1	TEST/RESERVED	RSLT[0:3]
1	0	0	0	0	AN0	RSLT[0:7]
1	0	0	0	1	AN1	RSLT[0:7]
1	0	0	1	0	AN2	RSLT[0:7]
1	0	0	1	1	AN3	RSLT[0:7]
1	0	1	0	0	AN4	RSLT[0:7]
1	0	1	0	1	AN5	RSLT[0:7]
1	0	1	1	0	AN6	RSLT[0:7]
1	0	1	1	1	AN7	RSLT[0:7]
1	1	0	0	0	RESERVED	RSLT[0:7]
1	1	0	0	1	RESERVED	RSLT[0:7]
1	1	0	1	0	RESERVED	RSLT[0:7]
1	1	0	1	1	RESERVED	RSLT[0:7]
1	1	1	0	0	V_{RH}	RSLT[0:7]
1	1	1	0	1	V_{RL}	RSLT[0:7]
1	1	1	1	0	$(V_{RH} - V_{RL}) / 2$	RSLT[0:7]
1	1	1	1	1	TEST/RESERVED	RSLT[0:7]

Multichannel Conversions

S8CM	CD	CC	CB	CA	Input	Result Register
0	0	0	X	X	AN[0:3]	RSLT[0:3]
0	0	1	X	X	AN[4:7]	RSLT[0:3]
0	1	0	X	X	RESERVED	RSLT[0:3]
0	1	1	X	X	V_{RH}	RSLT0
					V_{RL}	RSLT1
					$(V_{RH} - V_{RL}) / 2$	RSLT2
					TEST/RESERVED	RSLT3
1	0	X	X	X	AN[0:7]	RSLT[0:7]
1	1	X	X	X	RESERVED	RSLT0
					RESERVED	RSLT1
					RESERVED	RSLT2
					RESERVED	RSLT3
					V_{RH}	RSLT4
					V_{RL}	RSLT5
					$(V_{RH} - V_{RL}) / 2$	RSLT6
					TEST/RESERVED	RSLT7

D.2.6 ADSTAT — ADC Status Register $YFF70E

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

ADSTAT is a read-only register that contains the sequence complete flag, the conversion counter, and a channel-converted flag for each of the input channels.

SCF — Sequence Complete Flag

0 = Sequence not complete

1 = Sequence complete

When SCAN = 0, SCF is set at the end of a conversion sequence. When SCAN = 1, SCF is set at the end of the first conversion sequence. SCF is cleared when converter activity is halted or restarted by a write to ADCTL0.

CCTR[2:0] — Conversion Counter

This field shows the content of the conversion counter pointer during a conversion sequence. The value is the number of the next result register to be written.

CCF[7:0] — Conversion Complete Flags

Each bit in this field corresponds to an A/D result register. A bit is set when conversion of the corresponding input is complete. It remains set until the result register is read. It is cleared when the register is read.

D.2.7 RSLT[0:7] — ADC Result Registers $YFF710–$YFF73E

Result registers contain conversion results. Data format depends on the address from which it is read. The notation 10 in a diagram indicates that a bit is used only for 10-bit resolution and is cleared during 8-bit conversion. The notation 8/10 indicates a bit is used for both 8-bit and 10-bit resolution. Unused bits return zeros when read.

D.2.7.1 RJURR — Unsigned Right-Justified Result Registers $YFF710–$YFF71F

15						10	9	8	7	6	5	4	3	2	1	0
NOT USED							10	10	8/10	8/10	8/10	8/10	8/10	8/10	8/10	8/10

D.2.7.2 LJSRR — Signed Left-Justified Result Registers $YFF720–$YFF72F

15	14	13	12	11	10	9	8	7	6	5						0
8/10	8/10	8/10	8/10	8/10	8/10	8/10	8/10	10	10	NOT USED						

This data format assumes that the zero reference point is $(V_{RH} - V_{RL}) / 2$. Bit 15 thus indicates the sign of the result. When bit 15 = 1, the result is positive; when bit 15 = 0, the result is negative. Bits [5:0] return zeros when read.

D.2.7.3 LJURR — Unsigned Left-Justified Result Registers $YFF730–$YFF73F

15	14	13	12	11	10	9	8	7	6	5						0
8/10	8/10	8/10	8/10	8/10	8/10	8/10	8/10	10	10	NOT USED						

D

Table D–3. GPT Address Map

Address	15 8	7 0
$YFF900	GPT MODULE CONFIGURATION (GPTMCR)	
$YFF902	(RESERVED FOR TEST)	
$YFF904	INTERRUPT CONFIGURATION (ICR)	
$YFFE06	PGP DATA DIRECTION (DDRGP)	PGP DATA (PORTGP)
$YFF908	OC1 ACTION MASK (OC1M)	OC1 ACTION DATA (OC1D)
$YFF90A	TIMER COUNTER (TCNT)	
$YFF90C	PA CONTROL (PACTL)	PA COUNTER (PACNT)
$YFF90E	INPUT CAPTURE 1 (TIC1)	
$YFF910	INPUT CAPTURE 2 (TIC2)	
$YFF912	INPUT CAPTURE 3 (TIC3)	
$YFF914	OUTPUT COMPARE 1 (TOC1)	
$YFF916	OUTPUT COMPARE 2 (TOC2)	
$YFF918	OUTPUT COMPARE 3 (TOC3)	
$YFF91A	OUTPUT COMPARE 4 (TOC4)	
$YFF91C	INPUT CAPTURE 4/OUTPUT COMPARE 5 (TI4/O5)	
$YFF91E	TIMER CONTROL 1 (TCTL1)	TIMER CONTROL 2 (TCTL2)
$YFF920	TIMER MASK 1 (TMSK1)	TIMER MASK 2 (TMSK2)
$YFF922	TIMER FLAG 1 (TFLG1)	TIMER FLAG 2 (TFLG2)
$YFF924	FORCE COMPARE (CFORC)	PWM CONTROL C (PWMC)
$YFF926	PWM CONTROL A (PWMA)	PWM CONTROL B (PWMB)
$YFF928	PWM COUNT (PWMCNT)	
$YFF92A	PWMA BUFFER (PWMBUFA)	PWMB BUFFER (PWMBUFB)
$YFF92C	GPT PRESCALER (PRESCL)	
$YFF92E $YFF93F	RESERVED	

D.3.1 GPTMCR — GPT Module Configuration Register $YFF900

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP	FRZ1	FRZ0	STOPP	INCP	0	0	0	SUPV	0	0	0	IARB3	IARB2	IARB1	IARB0
RESET:															
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

GPTMCR bits control freeze, low-power stop, and single-step modes.

STOP — Stop Clocks

0 = Internal clocks not shut down

1 = Internal clocks shut down

FRZ[1:0] — FREEZE Response

FRZ1 is not used; FRZ0 encoding determines response to the IMB FREEZE signal.

0 = Ignore IMB FREEZE signal

1 = FREEZE the current state of the GPT

STOPP — Stop Prescaler

0 = Normal operation

1 = Stop prescaler and pulse accumulator from incrementing. Ignore changes to input pins.

INCP — Increment Prescaler

0 = Has no meaning

1 = If STOPP is asserted, increment prescaler once and clock input synchronizers once.

SUPV — Supervisor/Unrestricted Data Space

Because the CPU16 in the MC68HC16Z1 operates in supervisor mode only, this bit has no effect.

IARB[3:0] — 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.3.2 GPTMTR — GPT Module Test Register (Reserved) $YFF902

This address is currently unused and returns zeros when read. It is reserved for GPT factory test.

D.3.3 ICR — GPT Interrupt Configuration Register $YFF904

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

ICR fields determine internal and external interrupt priority, and provide the upper nibble of the interrupt vector number supplied to the CPU when an interrupt is acknowledged.

IPA — Interrupt Priority Adjust

Specifies which of the 11 internal GPT interrupt sources is assigned highest priority.

IPL — Interrupt Priority Level

Specifies the priority level of GPT interrupt requests.

IVBA — Interrupt Vector Base Address

Contains the most significant nibble of interrupt vector numbers supplied by the GPT.

D.3.4 DDRGP — Port GP Data Direction Register $YFF906

PORTGP — Port GP Data Register $YFF907

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

When GPT pins are used as an 8-bit port, DDRGP determines whether pins are input or output and PORTGP holds the 8-bit data.

DDRGP[7:0] — Parallel Data Direction Register

0 = Input only

1 = Output

PORTGP[7:0] — Parallel Data Register

D.3.5 OC1M — OC1 Action Mask Register $YFF908

OC1D — OC1 Action Data Register $YFF909

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC1M					0	0	0	OC1D					0	0	0
RESET															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

All OC outputs can be controlled by the action of OC1. OC1M contains a mask that determines which pins are affected, and OC1D determines what the outputs are.

OC1M[5:1] — OC1 Mask

0 = Corresponding output compare pin is not affected by OC1 compare.

1 = Corresponding output compare pin is affected by OC1 compare.

OC1M[5:1] correspond to OC[5:1].

OC1D[5:1] — OC1 Data

0 = If OC1 mask bit is set, clear corresponding output compare pin on OC1 match.

1 = If OC1 mask bit is set, set corresponding output compare pin on OC1 match.

OC1D[5:1] correspond to OC[5:1].

D.3.6 TCNT — Timer Counter Register $YFF90A

TCNT is the 16-bit free-running counter associated with the input capture, output compare, and pulse accumulator functions of the GPT module.

D.3.7 PACTL — Pulse Accumulator Control Register $YFF90C

PACNT — Pulse Accumulator Counter $YFF90D

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAIS	PAEN	PAMOD	PEDGE	PCLKS	I4/O5	PACLK		PULSE ACCUMULATOR COUNTER							
RESET:															
U	0	0	0	U	0	0	0	0	0	0	0	0	0	0	0

PACTL enables the pulse accumulator and selects either event counting or gated mode. In event counting mode, PACNT is incremented each time an event occurs. In gated mode, it is incremented by an internal clock.

PAIS — PAI Pin State (Read Only)

PAEN — Pulse Accumulator Enable
- 0 = Pulse accumulator disabled
- 1 = Pulse accumulator enabled

PAMOD — Pulse Accumulator Mode
- 0 = External event counting
- 1 = Gated time accumulation

PEDGE — Pulse Accumulator Edge Control

The effects of PEDGE and PAMOD are shown in the following table.

PAMOD	PEDGE	Effect
0	0	PAI Falling Edge Increments Counter
0	1	PAI Rising Edge Increments Counter
1	0	Zero on PAI Inhibits Counting
1	1	One on PAI Inhibits Counting

PCLKS — PCLK Pin State (Read Only)

I4/O5 — Input Capture 4/Output Compare 5
- 0 = Output compare 5 enabled
- 1 = Input capture 4 enabled

PACLK[1:0] — Pulse Accumulator Clock Select (Gated Mode)

PACLK[1:0]	Pulse Accumulator Clock Selected
00	System Clock Divided by 512
01	Same Clock Used to Increment TCNT
10	TOF Flag from TCNT
11	External Clock, PCLK

PACNT — Pulse Accumulator Counter

Eight-bit read/write counter used for external event counting or gated time accumulation.

D.3.8 TIC[1:3] — Input Capture Registers 1–3 $YFF90E–$YFF912

The input capture registers are 16-bit read-only registers which are used to latch the value of TCNT when a specified transition is detected on the corresponding input capture pin. They are reset to $FFFF.

D.3.9 TOC[1:4] — Output Compare Registers 1–4 $YFF914–$YFF91A

The output compare registers are 16-bit read/write registers which can be used as output waveform controls or as elapsed time indicators. For output compare functions, they are written to a desired match value and compared against TCNT to control specified pin actions. They are reset to $FFFF.

D.3.10 TI4/O5 — Input Capture 4/Output Compare 5 Register $YFF91C

This register serves either as input capture register 4 or output compare register 5, depending on the state of I4/O5 in PACTL.

D.3.11 TCTL1/TCTL2 — Timer Control Registers 1 and 2 $YFF91E

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OM5	OL5	OM4	OL4	OM3	OL3	OM2	OL2	EDGE4		EDGE3		EDGE2		EDGE1	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TCTL1 determines output compare mode and output logic level. TCTL2 determines the type of input capture to be performed.

OM/OL[5:2] — Output Compare Mode Bits and Output Compare Level Bits

Each pair of bits specifies an action to be taken when output comparison is successful.

OM/OL[5:2]	Action Taken
00	Timer Disconnected from Output Logic
01	Toggle OCx Output Line
10	Clear OCx Output Line to 0
11	Set OCx Output Line to 1

EDGE[4:1] — Input Capture Edge Control

Each pair of bits configures input sensing logic for the corresponding input capture.

EDGE[4:1]	Configuration
00	Capture Disabled
01	Capture on Rising Edge Only
10	Capture on Falling Edge Only
11	Capture on Any (Rising or Falling) Edge

D.3.12 TMSK1/TMSK2 — Timer Interrupt Mask Registers 1 and 2 $YFF920

15	14			11	10		8	7	6	5	4	3	2		0
I4/O5I	OCI				ICI			TOI	0	PAOVI	PAII	CPROUT	CPR		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMSK1 enables OC and IC interrupts. TMSK2 controls pulse accumulator interrupts and TCNT functions.

I4/O5I — Input Capture 4/Output Compare 5 Interrupt Enable
- 0 = IC4/OC5 interrupt disabled
- 1 = IC4/OC5 interrupt requested when I4/O5F flag in TFLG1 is set

OCI[4:1] — Output Compare Interrupt Enable
- 0 = OC interrupt disabled
- 1 = OC interrupt requested when OC flag set

OCI[4:1] correspond to OC[4:1].

ICI[3:1] — Input Capture Interrupt Enable
- 0 = IC interrupt disabled
- 1 = IC interrupt requested when IC flag set

ICI[3:1] correspond to IC[3:1].

TOI — Timer Overflow Interrupt Enable
- 0 = Timer overflow interrupt disabled
- 1 = Interrupt requested when TOF flag is set

PAOVI — Pulse Accumulator Overflow Interrupt Enable
- 0 = Pulse accumulator overflow interrupt disabled
- 1 = Interrupt requested when PAOVF flag is set

PAII — Pulse Accumulator Input Interrupt Enable
- 0 = Pulse accumulator interrupt disabled
- 1 = Interrupt requested when PAIF flag is set

CPROUT — Compare/Capture Unit Clock Output Enable
- 0 = Normal operation for OC1 pin
- 1 = TCNT clock driven out OC1 pin

CPR[2:0] — Timer Prescaler/PCLK Select Field

This field selects one of seven prescaler taps or PCLK to be TCNT input.

CPR[2:0]	System Clock Divide-by Factor
000	4
001	8
010	16
011	32
100	64
101	128
110	256
111	PCLK

D.3.13 TFLG1/TFLG2 — Timer Interrupt Flag Registers 1and 2 $YFF922

15	14			11	10			8	7	6	5	4	3	2	1	0
I4/O5F	OCF				ICF				TOF	0	PAOVF	PAIF	0	0	0	0
RESET:																
0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

These registers show condition flags that correspond to various GPT events. If the corresponding interrupt enable bit in TMSK1/TMSK2 is set, an interrupt will occur.

I4/O5F — Input Capture 4/Output Compare 5 Flag

When I4/O5 in PACTL is zero, this flag is set each time TCNT matches the value in TOC5. When I4/O5 in PACTL is one, the flag is set each time a selected edge is detected at the I4/O5 pin.

OCF[4:1] — Output Compare Flags

An output compare flag is set each time TCNT matches the corresponding TOC register. OCF[4:1] correspond to OC[4:1].

ICF[3:1] — Input Capture Flags

A flag is set each time a selected edge is detected at the corresponding input capture pin. ICF[3:1] correspond to IC[3:1].

TOF — Timer Overflow Flag

This flag is set each time TCNT advances from a value of $FFFF to $0000.

PAOVF — Pulse Accumulator Overflow Flag

This flag is set each time the pulse accumulator counter advances from a value of $FF to $00.

PAIF — Pulse Accumulator Flag

In event counting mode, this flag is set when an active edge is detected on the PAI pin. In gated time accumulation mode, it is set at the end of the timed period.

D.3.14 CFORC — Compare Force Register $YFF924

PWMC — PWM Control Register C **$YFF925**

15					11	10	9	8	7	6			4	3	2	1	0
FOC						0	FPWMA	FPWMB	PPROUT	PPR				SFA	SFB	F1A	F1B
RESET:																	
0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0

Setting a bit in CFORC will cause a specific output on OC or PWM pins. PWMC sets PWM operating conditions.

FOC[5:1] — Force Output Compare
- 0 = Has no meaning
- 1 = Causes pin action programmed for corresponding OC pin, but the OC flag is not set.

FOC[5:1] correspond to OC[5:1].

FPWMA — Force PWMA Value
- 0 = Normal PWMA operation
- 1 = The value of F1A is driven out on the PWMA pin, regardless of the state of PPROUT.

FPWMB — Force PWMB Value
- 0 = Normal PWMB operation
- 1 = The value of F1B is driven out on the PWMB pin.

PPROUT — PWM Clock Output Enable
- 0 = Normal PWM operation on PMWA
- 1 = TCNT clock driven out PWMA pin.

PPR[2:0] — PWM Prescaler/PCLK Select
This field selects one of seven prescaler taps or PCLK to be PWMCNT input.

PPR[2:0]	System Clock Divide-by Factor
000	2
001	4
010	8
011	16
100	32
101	64
110	128
111	PCLK

SFA — PWMA Slow/Fast Select
- 0 = PWMA period is 256 PWMCNT increments long.
- 1 = PWMA period is 32768 PWMCNT increments long.

D

SFB — PWMB Slow/Fast Select

0 = PWMB period is 256 PWMCNT increments long.

1 = PWMB period is 32768 PWMCNT increments long.

The following table shows the effects of SF settings on PWM frequency (16.78-MHz system clock).

PPR[2:0]	Prescaler Tap	SFA/B = 0	SFA/B = 1
000	Div 2 = 8.39 MHz	32.8 kHz	256 Hz
001	Div 4 = 4.19 MHz	16.4 kHz	128 Hz
010	Div 8 = 2.10 MHz	8.19 kHz	64.0 Hz
011	Div 16 = 1.05 MHz	4.09 kHz	32.0 Hz
100	Div 32 = 524 kHz	2.05 kHz	16.0 Hz
101	Div 64 = 262 kHz	1.02 kHz	8.0 Hz
110	Div 128 = 131 kHz	512 Hz	4.0 Hz
111	PCLK	PCLK/256	PCLK/32768

F1A — Force Logic Level One on PWMA

0 = Force logic level zero output on PWMA pin.

1 = Force logic level one output on PWMA pin.

F1B — Force Logic Level One on PWMB

0 = Force logic level zero output on PWMB pin.

1 = Force logic level one output on PWMB pin.

D.3.15 PWMA/PWMB — PWM Registers A/B $YFF926, $YFF927

The value in these registers determines pulse-width of the corresponding PWM output. A value of $00 corresponds to continuously low output; a value of $80 to 50% duty cycle. Maximum value ($FF) selects an output which is high for 255/256 of the period. Writes to these registers are buffered by PWMBUFA and PWMBUFB.

D.3.16 PWMCNT — PWM Count Register $YFF928

PWMCNT is the 16-bit free-running counter used for GPT PWM functions.

D.3.17 PWMBUFA — PWM Buffer Register A $YFF92A

PWMBUFB — PWM Buffer Register B $YFF92B

In order to prevent glitches when PWM duty cycle is changed, the contents of PWMA and PWMB are transferred to these read-only registers at the end of each duty cycle. Reset state is $0000.

D.3.18 PRESCL — GPT Prescaler $YFF92C

The 9-bit prescaler value can be read from bits [8:0] at this address. Bits [15:9] always read as zeros. Reset state is $0000.

Table D–4. SIM Address Map

Address	15 8	7 0
YFFA00	MODULE CONFIGURATION (SIMCR)	
YFFA02	FACTORY TEST (SIMTR)	
YFFA04	CLOCK SYNTHESIZER CONTROL (SYNCR)	
YFFA06	NOT USED	RESET STATUS (RSR)
YFFA08	MODULE TEST E (SIMTRE)	
YFFA0A	NOT USED	NOT USED
YFFA0C	NOT USED	NOT USED
YFFA0E	NOT USED	NOT USED
YFFA10	NOT USED	PORTE DATA (PORTE0)
YFFA12	NOT USED	PORTE DATA (PORTE1)
YFFA14	NOT USED	PORTE DATA DIRECTION (DDRE)
YFFA16	NOT USED	PORTE PIN ASSIGNMENT (PEPAR)
YFFA18	NOT USED	PORTF DATA (PORTF0)
YFFA1A	NOT USED	PORTF DATA (PORTF1)
YFFA1C	NOT USED	PORTF DATA DIRECTION (DDRF)
YFFA1E	NOT USED	PORTF PIN ASSIGNMENT (PFPAR)
YFFA20	NOT USED	SYSTEM PROTECTION CONTROL (SYPCR)
YFFA22	PERIODIC INTERRUPT CONTROL (PICR)	
YFFA24	PERIODIC INTERRUPT TIMING (PITR)	
YFFA26	NOT USED	SOFTWARE SERVICE (SWSR)
YFFA28	NOT USED	NOT USED
YFFA2A	NOT USED	NOT USED
YFFA2C	NOT USED	NOT USED
YFFA2E	NOT USED	NOT USED
YFFA30	TEST MODULE MASTER SHIFT A (TSTMSRA)	
YFFA32	TEST MODULE MASTER SHIFT B (TSTMSRB)	
YFFA34	TEST MODULE SHIFT COUNT (TSTSC)	
YFFA36	TEST MODULE REPETITION COUNTER (TSTRC)	
YFFA38	TEST MODULE CONTROL (CREG)	
YFFA3A	TEST MODULE DISTRIBUTED (DREG)	
YFFA3C	NOT USED	NOT USED
YFFA3E	NOT USED	NOT USED
YFFA40	NOT USED	PORT C DATA (PORTC)
YFFA42	NOT USED	NOT USED

Table D–4. SIM Address Map (Continued)

Address	15 8	7 0
YFFA44	CHIP-SELECT PIN ASSIGNMENT (CSPAR0)	
YFFA46	CHIP-SELECT PIN ASSIGNMENT (CSPAR1)	
YFFA48	CHIP-SELECT BASE BOOT (CSBARBT)	
YFFA4A	CHIP-SELECT OPTION BOOT (CSORBT)	
YFFA4C	CHIP-SELECT BASE 0 (CSBAR0)	
YFFA4E	CHIP-SELECT OPTION 0 (CSOR0)	
YFFA50	CHIP-SELECT BASE 1 (CSBAR1)	
YFFA52	CHIP-SELECT OPTION 1 (CSOR1)	
YFFA54	CHIP-SELECT BASE 2 (CSBAR2)	
YFFA56	CHIP-SELECT OPTION 2 (CSOR2)	
YFFA58	CHIP-SELECT BASE 3 (CSBAR3)	
YFFA5A	CHIP-SELECT OPTION 3 (CSOR3)	
YFFA5C	CHIP-SELECT BASE 4 (CSBAR4)	
YFFA5E	CHIP-SELECT OPTION 4 (CSOR4)	
YFFA60	CHIP-SELECT BASE 5 (CSBAR5)	
YFFA62	CHIP-SELECT OPTION 5 (CSOR5)	
YFFA64	CHIP-SELECT BASE 6 (CSBAR6)	
YFFA66	CHIP-SELECT OPTION 6 (CSOR6)	
YFFA68	CHIP-SELECT BASE 7 (CSBAR7)	
YFFA6A	CHIP-SELECT OPTION 7 (CSOR7)	
YFFA6C	CHIP-SELECT BASE 8 (CSBAR8)	
YFFA6E	CHIP-SELECT OPTION 8 (CSOR8)	
YFFA70	CHIP-SELECT BASE 9 (CSBAR9)	
YFFA72	CHIP-SELECT OPTION 9 (CSOR9)	
YFFA74	CHIP-SELECT BASE 10 (CSBAR10)	
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

D.4.1 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. Can be read or written at any time, except for the module mapping (MM) bit, which can be written once, and must remain set.

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/User Data Space

The MC68HC16Z1 operates only in supervisory mode — SUPV has no effect.

MM — Module Mapping

Because ADDR[23:20] follow the state of ADDR19 in the CPU16, MM must remain set.

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

D.4.2 SIMTR — System Integration Test Register $YFFA02

SIMTR is used for factory test only.

D.4.3 SYNCR — Clock Synthesizer Control Register $YFFA04

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	X	Y						EDIV	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:

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

W — Frequency Control (VCO)
- 0 = Base VCO frequency.
- 1 = VCO frequency multiplied by four.

X — Frequency Control Bit (Prescale)
- 0 = Base system clock frequency
- 1 = System clock frequency multiplied by two.

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.4.4 RSR — Reset Status Register $YFFA07

15	14	13	12	11	10	9	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 CPU RESET instruction. Not used by MC68HC16Z1. The CPU16 has no reset instruction.

TST — Test Submodule Reset

Reset caused by the test submodule. Used during system test only.

D.4.5 SIMTRE — System Integration Test Register (ECLK) $YFFA08

Register is used for factory test only.

D.4.6 PORTE0/PORTE1 — Port E Data Register $YFFA11, $YFFA13

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

D

PORTE 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 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. Reads of PE3 always return one.

D.4.7 DDRE — Port E Data Direction Register $YFFA15

15	14	13	12	11	10	9	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.4.8 PEPAR — Port E Pin Assignment Register $YFFA17

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								PE7 (SIZ1)	PE6 (SIZ0)	PE5 (AS)	PE4 (DS)	PE3	PE2 (AVEC)	PE1 DSACK1	PE0 DSACK0
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. PE3 is not connected to a pin. DDE3, PE3, and PEPAR3 can be read and written, but have no function.

D.4.9 PORTF0/PORTF1— Port F Data Register $YFFA19, $YFFA1B

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

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.4.10 DDRF — Port F Data Direction Register $YFFA1D

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

D

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.4.11 PFPAR — Port F Pin Assignment Register $YFFA1F

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								PF7 (IRQ7)	PF6 (IRQ6)	PF5 (IRQ5)	PF4 (IRQ4)	PF3 (IRQ3)	PF2 (IRQ2)	PF1 (IRQ1)	PF0 (MODCLK)
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 bus control signal; clearing a bit assigns the pin to I/O port F.

D.4.12 SYPCR — System Protection Control Register $YFFA21

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								SWE	SWP	SWT		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 timeout 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 timeout period.

Bus Monitor Period

BMT	Bus Monitor Timeout Period
00	64 System Clocks
01	32 System Clocks
10	16 System Clocks
11	8 System Clocks

D.4.13 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 periodic interrupt vector number supplied by the SIM when the CPU acknowledges an interrupt request.

D.4.14 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
1 = Periodic timer clock prescaled by a value of 512
0 = Periodic timer clock not prescaled

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.

PIT Period = [(PIT Modulus)(Prescaler value)(4)]/EXTAL Frequency

D.4.15 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.4.16 TSTMSRA — Master Shift Register A $YFFA30

Register is used for factory test only.

D.4.17 TSTMSRB — Master Shift Register B $YFFA32

Register is used for factory test only.

D.4.18 TSTSC — Test Module Shift Count $YFFA34

Register is used for factory test only.

D.4.19 TSTRC — Test Module Repetition Count $YFFA36

Register is used for factory test only.

D.4.20 CREG — Test Submodule Control Register $YFFA38

Register is used for factory test only.

D.4.21 DREG — Distributed Register $YFFA3A

Register is used for factory test only.

D.4.22 PORTC — Port C Data Register $YFFA41

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								0	DO6	DO5	DO4	DO3	DO2	DO1	DO0
RESET:								0	1	1	1	1	1	1	1

PORTC latches data for chip-select pins that are used for discrete output.

D

D.4.23 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	$\overline{CS5}$ (FC2)	$\overline{CS4}$ (FC1)	$\overline{CS3}$ (FC0)	$\overline{CS2}$ (BGACK)	$\overline{CS1}$ (BG)	$\overline{CS0}$ (BR)	$\overline{CSBOOT}$
RESET:								
0	0	DATA2 1	DATA2 1	DATA2 1	DATA1 1	DATA1 1	DATA1 1	1 DATA 0

Contains seven 2-bit fields ($\overline{CS[5:0]}$ and $\overline{CSBOOT}$) 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. Mnemonics in parentheses show alternate functions that can be enabled by data bus mode selection during reset.

D.4.24 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	$\overline{CS10}$ (ADDR23)	$\overline{CS9}$ (ADDR22)	$\overline{CS8}$ (ADDR21)	$\overline{CS7}$ (ADDR20)	$\overline{CS6}$ (ADDR19)
RESET:										
0	0	0	0	0	0	DATA7 1	DATA6 1	DATA5 1	DATA4 1	DATA3 1

Contains five 2-bit fields ($\overline{CS[10:6]}$) that determine the functions of corresponding chip-select pins. CSPAR1[15:10] are not used. These bits always read zero; write has no effect. Mnemonics in parentheses show 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.4.25 CSBARBT — Chip Select Base Address Register Boot ROM $YFFA48

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

D.4.26 CSBAR[0:10] — Chip Select Base Address Registers $YFFA4C–$YFFA74

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	BLKSZ		
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. ADDR[23:20] are at the same logic level as ADDR19 during normal operation. 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[15:3] — 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	ADDR23–ADDR11
001	8 K	ADDR23–ADDR13
010	16 K	ADDR23–ADDR14
011	64 K	ADDR23–ADDR16
100	128 K	ADDR23–ADDR17
101	256 K	ADDR23–ADDR18
110	512 K	ADDR23–ADDR19
111	512 K	ADDR23–ADDR20

D.4.27 CSORBT — Chip Select Option Register Boot ROM $YFFA4A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE	BYTE		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.4.28 CSOR[0:10] — Chip Select Option Registers $YFFA4E–$YFFA76

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

1 = Data strobe

0 = Address strobe

$\overline{DSACK}$ — Data Strobe Acknowledge

This field specifies the source of $\overline{DSACK}$ in asynchronous 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.

$\overline{AVEC}$ — Autovector Enable

Do not enable autovector support when in synchronous mode.

1 = Autovector enabled

0 = External interrupt vector enabled

Option Register Function Summary

MODE	BYTE	R/$\overline{W}$	STRB	$\overline{DSACK}$	SPACE	IPL	$\overline{AVEC}$
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			

D

Table D–5. SRAM Address Map

Address	15 8	7 0
$YFFB00	RAM MODULE CONFIGURATION REGISTER (RAMMCR)	
$YFFB02	RAM TEST REGISTER (RAMTST)	
$YFFB04	RAM ARRAY BASE ADDRESS REGISTER HIGH (RAMBAH)	
$YFFB06	RAM ARRAY BASE ADDRESS REGISTER LOW (RAMBAL)	

D.5.1 RAMMCR — RAM Module Configuration Register $YFFB00

15				11		9	8	7	6	5	4	3	2	1	0
STOP	0	0	0	RLCK	0	RASP		NOT USED							
RESET:															
1	0	0	0	0	0	1	1								

RAMMCR is used to determine whether the RAM is in STOP mode or normal mode. It is also used to determine in which space the array resides, and controls access to the base array registers. Reads of unimplemented bits always return zeros. Writes do not affect unimplemented bits.

STOP — Stop Control

0 = RAM array operates normally.

1 = RAM array enters low-power stop mode.

This bit determines whether the RAM array is in low-power stop mode. Reset state is one, leaving the array configured for low-power stop operation. In stop mode, the array retains its contents, but cannot be read or written by the CPU. This bit can be read or written at any time.

RLCK — RAM Base Address Lock

0 = SRAM base address registers can be written from IMB

1 = SRAM base address registers are locked

RLCK defaults to zero on reset; it can be written once to one.

RASP[1:0] — RAM Array Space

RASP limits access to the SRAM array in microcontrollers that support separate user and supervisor operating modes. Because the CPU16 operates in supervisor mode only, RASP1 has no effect.

RASP	Space
X0	Program and Data
X1	Program

D

D.5.2 RAMTST — RAM Test Register $YFFB02

RAMTST is used for factory test only. Reads of this register return zeros, and writes have no effect.

D.5.3 RAMBAH — Array Base Address Register High $YFFB04

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

D.5.4 RAMBAL — Array Base Address Register Low $YFFB06

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

RAMBAH and RAMBAL are used to specify an SRAM array base address. RAMBAH and RAMBAL can only be written while the SRAM is in low-power mode (RAMMCR STOP = 1) and the base address lock (RAMMCR RLCK = 0) is disabled. This prevents accidental remapping of the array.

Table D–6. QSM Address Map

Address	15 8	7 0
$YFFC00	QSM MODULE CONFIGURATION (QSMCR)	
$YFFC02	QSM TEST (QTEST)	
$YFFC04	QSM INTERRUPT LEVEL (QUILR)	QSM INTERRUPT VECTOR (QIVR)
$YFFC06	RESERVED	
$YFFC08	SCI CONTROL 0 (SCCR0)	
$YFFC0A	SCI CONTROL 1 (SCCR1)	
$YFFC0C	SCI STATUS (SCSR)	
$YFFC0E	SCI DATA (SCDR)	
$YFFC10	RESERVED	
$YFFC12	RESERVED	
$YFFC14	RESERVED	PQS DATA (PORTQS)
$YFFC16	PQS PIN ASSIGNMENT (PQSPAR)	PQS DATA DIRECTION (DDRQS)
$YFFC18	SPI CONTROL 0 (SPCR0)	
$YFFC1A	SPI CONTROL 1 (SPCR1)	
$YFFC1C	SPI CONTROL 2 (SPCR2)	
$YFFC1E	SPI CONTROL 3 (SPCR3)	SPI STATUS (SPSR)
$YFFC20–$YFFCFF	RESERVED	
$YFFD00–$YFFD1F	RECEIVE RAM (RR[0:F])	
$YFFD20–$YFFD3F	TRANSMIT RAM (TR[0:F])	
$YFFD40–$YFFD4F	COMMAND RAM (CR[0:F])	

D.6.1 QMCR — QSM Configuration Register $YFFC00

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

QMCR 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 QMCR 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.

Bits [12:8] — Not Implemented

SUPV — Supervisor/Unrestricted

Because the CPU16 in the MC68HC16Z1 operates only in supervisor mode, this bit has no effect.

Bits [6:4] — Not Implemented

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.6.2 QTEST — QSM Test Register $YFFC02

Used for factory test only.

D.6.3 QILR — QSM Interrupt Level Register $YFFC04

QIVR — QSM Interrupt Vector Register $YFFC05

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

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 CPU16 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 CPU16 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.6.4 PORTQS — Port QS Data Register $YFFC15

15 … 8	7	6	5	4	3	2	1	0
RESERVED	PQS7	PQS6	PQS5	PQS4	PQS3	PQS2	PQS1	PQS0
	RESET:							
	0	0	0	0	0	0	0	0

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

D.6.5 PQSPAR — Pin Assignment Register $YFFC16

DDRQS — Data Direction Register $YFFC17

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PQSPAR								DDRQS							
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. 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. 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.6.6 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	BR							
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). At reset, BAUD is initialized to a 2.1-MHz SCK frequency.

D.6.7 SPCR1 — QSPI Control Register 1 $YFFC1A

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

SPCR1 enables the QSPI and specified transfer delays. The CPU 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 may 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.6.8 SPCR2 — QSPI Control Register 2 $YFFC1C

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

Bit 12 — Not Implemented

ENDQP — Ending Queue Pointer
This field contains the last QSPI queue address.

Bits [7:4] — Not Implemented

NEWQP — New Queue Pointer Value
This field contains the first QSPI queue address.

D.6.9 SPCR3 — QSPI Control Register 3 $YFFC1E

SPSR — QSPI Status Register $YFFC1F

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

Bits [15:11] — Not Implemented

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

Bit 4 — Not Implemented

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.6.10 RR[0:F] — Receive Data Ram $YFFD00–$YFFD0E

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.

D.6.11 TR[0:F] — Transmit Data Ram $YFFD20–$YFFD3E

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 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.6.12 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 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 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 may 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 QPDR after transfer is complete.
1 = Peripheral chip selects remain asserted after transfer is complete.

BITSE — Bits per Transfer Enable

0 = 8 bits
1 = Number of bits set in BITS 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.6.13 SCCR0 — SCI Control Register 0 $YFFC08

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	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 CPU can read and write SCCR0 at any time. Changing the value of SCCR0 bits during a transfer operation can disrupt operation.

Bits [15:13] — Not Implemented

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:

$$\text{SCI Baud Clock Rate} = \text{System Clock}/(32\text{SCBR})$$

D.6.14 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 SCCR0 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.

Bit 15 — Not Implemented

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 may 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.6.15 SCSR — SCI Status Register $YFFC0C

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

0 = No noise detected on the received data.

1 = Noise occurred on the received data.

FE — Framing Error

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

0 = No framing error on the received data.

PF — Parity Error

1 = Parity error occurred on the received data.

0 = No parity error on the received data.

D.6.16 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 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 9-bit operation. When it is configured for 8-bit operation, they have no meaning or effect.

Table D–7. MC68HC16Z1 Module Address Map

(Assumes SIMCR MM = 1)

ADC		
Address	**15 8**	**7 0**
$FFF700	MODULE CONFIGURATION (ADCMCR)	
$FFF702	FACTORY TEST (ADTEST)	
$FFF704	(RESERVED)	
$FFF706	PORTADA DATA (PORTADA)	
$FFF708	(RESERVED)	
$FFF70A	ADC CONTROL 0 (ADCTL0)	
$FFF70C	ADC CONTROL 1 (ADCTL1)	
$FFF70E	ADC STATUS (ADSTAT)	
$FFF710	RIGHT-JUSTIFIED UNSIGNED RESULT 0 (RJURR0)	
$FFF712	RIGHT-JUSTIFIED UNSIGNED RESULT 1 (RJURR1)	
$FFF714	RIGHT-JUSTIFIED UNSIGNED RESULT 2 (RJURR2)	
$FFF716	RIGHT-JUSTIFIED UNSIGNED RESULT 3 (RJURR3)	
$FFF718	RIGHT-JUSTIFIED UNSIGNED RESULT 4 (RJURR4)	
$FFF71A	RIGHT-JUSTIFIED UNSIGNED RESULT 5 (RJURR5)	
$FFF71C	RIGHT-JUSTIFIED UNSIGNED RESULT 6 (RJURR6)	
$FFF71E	RIGHT-JUSTIFIED UNSIGNED RESULT 7 (RJURR7)	
$FFF720	LEFT-JUSTIFIED SIGNED RESULT 0 (LJSRR0)	
$FFF722	LEFT-JUSTIFIED SIGNED RESULT 1 (LJSRR1)	
$FFF724	LEFT-JUSTIFIED SIGNED RESULT 2 (LJSRR2)	
$FFF726	LEFT-JUSTIFIED SIGNED RESULT 3 (LJSRR3)	
$FFF728	LEFT-JUSTIFIED SIGNED RESULT 4 (LJSRR4)	
$FFF72A	LEFT-JUSTIFIED SIGNED RESULT 5 (LJSRR5)	
$FFF72C	LEFT-JUSTIFIED SIGNED RESULT 6 (LJSRR6)	
$FFF72E	LEFT-JUSTIFIED SIGNED RESULT 7 (LJSRR7)	
$FFF730	LEFT-JUSTIFIED UNSIGNED RESULT 0 (LJURR0)	
$FFF732	LEFT-JUSTIFIED UNSIGNED RESULT 1 (LJURR1)	
$FFF734	LEFT-JUSTIFIED UNSIGNED RESULT 2 (LJURR2)	
$FFF736	LEFT-JUSTIFIED UNSIGNED RESULT 3 (LJURR3)	
$FFF738	LEFT-JUSTIFIED UNSIGNED RESULT 4 (LJURR4)	
$FFF73A	LEFT-JUSTIFIED UNSIGNED RESULT 5 (LJURR5)	
$FFF73C	LEFT-JUSTIFIED UNSIGNED RESULT 6 (LJURR6)	
$FFF73E	LEFT-JUSTIFIED UNSIGNED RESULT 7 (LJURR7)	

GPT		
Address	**15 8**	**7 0**
$FFF900	GPT MODULE CONFIGURATION (GPTMCR)	
$FFF902	(RESERVED FOR TEST)	
$FFF904	INTERRUPT CONFIGURATION (ICR)	
$FFFE06	PGP DATA DIRECTION (DDRGP)	PGP DATA (PORTGP)
$FFF908	OC1 ACTION MASK (OC1M)	OC1 ACTION DATA (OC1D)
$FFF90A	TIMER COUNTER (TCNT)	
$FFF90C	PA CONTROL (PACTL)	PA COUNTER (PACNT)
$FFF90E	INPUT CAPTURE 1 (TIC1)	
$FFF910	INPUT CAPTURE 2 (TIC2)	
$FFF912	INPUT CAPTURE 3 (TIC3)	
$FFF914	OUTPUT COMPARE 1 (TOC1)	
$FFF916	OUTPUT COMPARE 2 (TOC2)	
$FFF918	OUTPUT COMPARE 3 (TOC3)	
$FFF91A	OUTPUT COMPARE 4 (TOC4)	
$FFF91C	INPUT CAPTURE 4/OUTPUT COMPARE 5 (TI4/O5)	
$FFF91E	TIMER CONTROL 1 (TCTL1)	TIMER CONTROL 2 (TCTL2)
$FFF920	TIMER MASK 1 (TMSK1)	TIMER MASK 2 (TMSK2)
$FFF922	TIMER FLAG 1 (TFLG1)	TIMER FLAG 2 (TFLG2)
$FFF924	FORCE COMPARE (CFORC)	PWM CONTROL C (PWMC)
$FFF926	PWM CONTROL A (PWMA)	PWM CONTROL B (PWMB)
$FFF928	PWM COUNT (PWMCNT)	
$FFF92A	PWMA BUFFER (PWMBUFA)	PWMB BUFFER (PWMBUFB)
$FFF92C	GPT PRESCALER (PRESCL)	
$FFF92E $FFF93F	RESERVED	

SIM		
Address	**15 8**	**7 0**
$FFFA00	MODULE CONFIGURATION (SIMCR)	
$FFFA02	FACTORY TEST (SIMTR)	
$FFFA04	CLOCK SYNTHESIZER CONTROL (SYNCR)	
$FFFA06	NOT USED	RESET STATUS (RSR)
$FFFA08	MODULE TEST E (SIMTRE)	
$FFFA0A	NOT USED	NOT USED
$FFFA0C	NOT USED	NOT USED
$FFFA0E	NOT USED	NOT USED
$FFFA10	NOT USED	PORTE DATA (PORTE0)
$FFFA12	NOT USED	PORTE DATA (PORTE1)
$FFFA14	NOT USED	PORTE DATA DIRECTION (DDRE)
$FFFA16	NOT USED	PORTE PIN ASSIGNMENT (PEPAR)
$FFFA18	NOT USED	PORTF DATA (PORTF0)
$FFFA1A	NOT USED	PORTF DATA (PORTF1)
$FFFA1C	NOT USED	PORTF DATA DIRECTION (DDRF)
$FFFA1E	NOT USED	PORTF PIN ASSIGNMENT (PFPAR)
$FFFA20	NOT USED	SYSTEM PROTECTION CONTROL (SYPCR)
$FFFA22	PERIODIC INTERRUPT CONTROL (PICR)	
$FFFA24	PERIODIC INTERRUPT TIMING (PITR)	
$FFFA26	NOT USED	SOFTWARE SERVICE (SWSR)
$FFFA28	NOT USED	NOT USED
$FFFA2A	NOT USED	NOT USED
$FFFA2C	NOT USED	NOT USED
$FFFA2E	NOT USED	NOT USED
$FFFA30	TEST MODULE MASTER SHIFT A (TSTMSRA)	
$FFFA32	TEST MODULE MASTER SHIFT B (TSTMSRB)	
$FFFA34	TEST MODULE SHIFT COUNT (TSTSC)	
$FFFA36	TEST MODULE REPETITION COUNTER (TSTRC)	
$FFFA38	TEST MODULE CONTROL (CREG)	
$FFFA3A	TEST MODULE DISTRIBUTED (DREG)	
$FFFA3C	NOT USED	NOT USED
$FFFA3E	NOT USED	NOT USED
$FFFA40	NOT USED	PORT C DATA (CSPDR)
$FFFA42	NOT USED	NOT USED

SIM (CONTINUED)		
Address	**15 8**	**7 0**
$FFFA44	CHIP-SELECT PIN ASSIGNMENT (CSPAR0)	
$FFFA46	CHIP-SELECT PIN ASSIGNMENT (CSPAR1)	
$FFFA48	CHIP-SELECT BASE BOOT (CSBARBT)	
$FFFA4A	CHIP-SELECT OPTION BOOT (CSORBT)	
$FFFA4C	CHIP-SELECT BASE 0 (CSBAR0)	
$FFFA4E	CHIP-SELECT OPTION 0 (CSOR0)	
$FFFA50	CHIP-SELECT BASE 1 (CSBAR1)	
$FFFA52	CHIP-SELECT OPTION 1 (CSOR1)	
$FFFA54	CHIP-SELECT BASE 2 (CSBAR2)	
$FFFA56	CHIP-SELECT OPTION 2 (CSOR2)	
$FFFA58	CHIP-SELECT BASE 3 (CSBAR3)	
$FFFA5A	CHIP-SELECT OPTION 3 (CSOR3)	
$FFFA5C	CHIP-SELECT BASE 4 (CSBAR4)	
$FFFA5E	CHIP-SELECT OPTION 4 (CSOR4)	
$FFFA60	CHIP-SELECT BASE 5 (CSBAR5)	
$FFFA62	CHIP-SELECT OPTION 5 (CSOR5)	
$FFFA64	CHIP-SELECT BASE 6 (CSBAR6)	
$FFFA66	CHIP-SELECT OPTION 6 (CSOR6)	
$FFFA68	CHIP-SELECT BASE 7 (CSBAR7)	
$FFFA6A	CHIP-SELECT OPTION 7 (CSOR7)	
$FFFA6C	CHIP-SELECT BASE 8 (CSBAR8)	
$FFFA6E	CHIP-SELECT OPTION 8 (CSOR8)	
$FFFA70	CHIP-SELECT BASE 9 (CSBAR9)	
$FFFA72	CHIP-SELECT OPTION 9 (CSOR9)	
$FFFA74	CHIP-SELECT BASE 10 (CSBAR10)	
$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

SRAM		
Address	**15 8**	**7 0**
$FFFB00	RAM MODULE CONFIGURATION REGISTER (RAMMCR)	
$FFFB02	RAM TEST REGISTER (RAMTST)	
$FFFB04	RAM ARRAY BASE ADDRESS REGISTER HIGH (RAMBAH)	
$FFFB06	RAM ARRAY BASE ADDRESS REGISTER LOW (RAMBAL)	
QSM		
Address	**15 8**	**7 0**
$FFFC00	QSM MODULE CONFIGURATION (QSMCR)	
$FFFC02	QSM TEST (QTEST)	
$FFFC04	QSM INTERRUPT LEVEL (QUILR)	QSM INTERRUPT VECTOR (QIVR)
$FFFC06	RESERVED	
$FFFC08	SCI CONTROL 0 (SCCR0)	
$FFFC0A	SCI CONTROL 1 (SCCR1)	
$FFFC0C	SCI STATUS (SCSR)	
$FFFC0E	SCI DATA (SCDR)	
$FFFC10	RESERVED	
$FFFC12	RESERVED	
$FFFC14	RESERVED	PQS DATA (PORTQS)
$FFFC16	PQS PIN ASSIGNMENT (PQSPAR)	PQS DATA DIRECTION (DDRQS)
$FFFC18	SPI CONTROL 0 (SPCR0)	
$FFFC1A	SPI CONTROL 1 (SPCR1)	
$FFFC1C	SPI CONTROL 2 (SPCR2)	
$FFFC1E	SPI CONTROL 3 (SPCR3)	SPI STATUS (SPSR)
$FFFC20–$FFFCFF	RESERVED	
$FFFD00–$FFFD1F	RECEIVE RAM (RR[0:F])	
$FFFD20–$FFFD3F	TRANSMIT RAM (TR[0:F])	
$FFFD40–$FFFD4F	COMMAND RAM (CR[0:F])	

Table D–8. Register Bit and Field Mnemonics

Mnemonic	Name	Register Location
ADDR[15:3]	Base Address	CSBAR[0:10], CSBARBT
$\overline{\text{AVEC}}$	Autovector Enable	CSOR[0:10], CSORBT
BITS	Bits Per Transfer	SPCR0
BITSE	Bits Per Transfer Enable	QSPI Command RAM
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
CCF[7:0]	Conversion Complete Flags	ADSTAT
CCTR[2:0]	Conversion Counter	ADSTAT
CONT	Continue	QSPI Command RAM
CPHA	Clock Phase	SPCR0
CPOL	Clock Polarity	SPCR0
CPROUT	Compare/Capture Unit Clock Output Enable	TMSK2
CPR[2:0]	Timer Prescaler/PCLK Select Field	TMSK2
CPTQP	Completed Queue Pointer	SPSR
CR[0:F]	Command RAM	QSPI RAM
CSOR[0:10]	Chip Select Option Registers	CSOR[0:10], CSORBT
$\overline{\text{DSACK}}$	Data Strobe Acknowledge	CSOR[0:10], CSORBT
DSCK	PCS to SCK Delay	QSPI Command RAM
DSCKL	Delay before SCK	SPCR1
DT	Delay after Transfer	QSPI Command RAM
DTL	Length of Delay after Transfer	SPCR1
EDGE[4:1]	Input Capture Edge Control	TCTL2
EDIV	ECLK Divide Rate	SYNCR
ENDQP	Ending Queue Pointer	SPCR2
EV	Extension Bit Overflow Flag	CCR
EXOFF	External Clock Off	SIMCR
EXT	External Reset	RSR
F1A	Force Logic Level One on PWMA	PWMC
F1B	Force Logic Level One on PWMB	PWMC
FE	Framing Error	SCSR
FOC[5:1]	Force Output Compare	CFORC
FPWMA	Force PWMA Value	CFORC
FPWMB	Force PWMB Value	CFORC

Table D–8. Register Bit and Field Mnemonics (Continued)

FRZBM	Freeze Bus Monitor Enable	SIMCR
FRZSW	Freeze Software Enable	SIMCR
FRZ[1:0]	Freeze	ADCMCR
FRZ[1:0]	FREEZE Response	GPTMCR
FRZ[1:0]	Freeze Control	QSMCR
H	Half Carry Flag	CCR
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
I4/O5	Input Capture 4/Output Compare 5	PACNT
I4/O5F	Input Capture 4/Output Compare 5 Flag	TFLG1
I4/O5I	I4/O5 Interrupt Enable	TMSK1
IARB	Interrupt Arbitration	QSMCR
IARB[3:0]	Interrupt Arbitration	GPTMCR
IARB[3:0]	Interrupt Arbitration Field	SIMCR
ICF[3:1]	Input Capture Flags	TFLG1
ICI[3:1]	Input Capture Interrupt Enable	TMSK1
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
INCP	Increment Prescaler	GPTMCR
INTV[7:0]	Interrupt Vector Number	QIVR
IPA	Interrupt Priority Adjust	ICR
IPL	Interrupt Priority Level	CSOR[0:10], CSORBT
IPL	Interrupt Priority Level	ICR
IP[2:0]	Interrupt Priority Field	CCR
IVBA	Interrupt Vector Base Address	ICR
LJSRR	Signed Left-Justified Result	RSLT[0:7]
LJURR	Unsigned Left-Justified Result	RSLT[0:7]
LOC	Loss of Clock Reset	RSR
LOOPQ	QSPI Loop Mode	SPCR3
LOOPS	Loop Mode	SCCR1
M	Mode Select	SCCR1
MISO	Master In Slave Out	DDRQS
MISO	Master In Slave Out	PQSPAR
MM	Module Mapping	SIMCR

Table D–8. Register Bit and Field Mnemonics (Continued)

MODE	Asynchronous/Synchronous Mode	CSOR[0:10], CSORBT
MODF	Mode Fault Flag	SPSR
MOSI	Master Out Slave In	DDRQS
MOSI	Master Out Slave In	PQSPAR
MSTR	Master/Slave Mode Select	SPCR0
MULT	Multichannel Conversion	ADCTL1
MV	Accumulator M Overflow Flag	CCR
N	Negative Flag	CCR
NEWQP	New Queue Pointer Value	SPCR2
NF	Noise Error	SCSR
OC1D[5:1]	OC1 Data	OC1D
OC1M[5:1]	OC1 Mask	OC1M
OCF[4:1]	Output Compare Flags	TFLG1
OCI[4:1]	Output Compare Interrupt Enable	TMSK1
OM[5:2]	Output Compare Mode Bits	TCTL1
OL[5:2]	Output Compare Level Bits	TCTL1
OR	Overrun Error	SCSR
PACLK[1:0]	Pulse Accumulator Clock Select (Gated Mode)	PACNT
PAEN	Pulse Accumulator Enable	PACNT
PAIF	Pulse Accumulator Flag	TFLG2
PAII	Pulse Accumulator Input Interrupt Enable	TMSK2
PAIS	PAI Pin State (Read Only)	PACNT
PAMOD	Pulse Accumulator Mode	PACNT
PAOVF	Pulse Accumulator Overflow Flag	TFLG2
PAOVI	Pulse Accumulator Overflow Interrupt Enable	TMSK2
PCLKS	PCLK Pin State (Read Only)	PACNT
PCS0/$\overline{SS}$	Peripheral Chip Select 0/Slave Select	PQSPAR
PCS[3:0]	Peripheral Chip Select	QSPI command RAM
PCS[3:1]	Peripheral Chip Selects	PQSPAR
PE	Parity Enable	SCCR1
PEDGE	Pulse Accumulator Edge Control	PACNT
PF	Parity Error	SCSR
PIRQL[2:0]	Periodic Interrupt Request Level	PICR
PITM[7:0]	Periodic Interrupt Timing Modulus	PITR
PIV[7:0]	Periodic Interrupt Vector	PICR
PK[3:0]	Program Counter Address Extension Field	CCR
PORTQS	Port QS Data Register	QIVR
POW	Power-Up Reset	RSR
PPROUT	PWM Clock Output Enable	PWMC
PPR[2:0]	PWM Prescaler/PCLK Select	PWMC

Table D–8. Register Bit and Field Mnemonics (Continued)

PRS[4:0]	Prescaler Rate Selection	ADCTL0
PT	Parity Type	SCCR1
PTP	Periodic Timer Prescaler Control	PITR
R/$\overline{W}$	Read/Write	CSOR[0:10], CSORBT
RAF	Receiver Active	SCSR
RASP[1:0]	RAM Array Space	RAMMCR
RDRF	Receive Data Register Full	SCSR
RE	Receiver Enable	SCCR1
RES10	10-Bit Resolution	ADCTL0
RIE	Receiver Interrupt Enable	SCCR1
RJURR	Unsigned Right-Justified Result	RSLT[0:7]
RLCK	RAM Base Address Lock	RAMMCR
RR[0:F]	Receive Data Ram	QSPI RAM
RSTEN	Reset Enable	SYNCR
RWU	Receiver Wakeup	SCCR1
S	STOP Enable	CCR
S8CM	Select Eight-Conversion Sequence Mode	ADCTL1
SBK	Send Break	SCCR1
SCAN	Scan Mode Selection	ADCTL1
SCBR	SCI Baud Rate	SCCR0
SCF	Sequence Complete Flag	ADSTAT
SCK	Serial Clock	DDRQS
SFA	PWMA Slow/Fast Select	PWMC
SFB	PWMB Slow/Fast Select	PWMC
SHEN[1:0]	Show Cycle Enable	SIMCR
SLIMP	Limp Mode	SYNCR
SLOCK	Synthesizer Lock	SYNCR
SLVEN	Factory Test Mode Enabled	SIMCR
SM	Saturate Mode Bit	CCR
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
STEXT	Stop Mode External Clock	SYNCR
STOP	STOP Mode	ADCMCR
STOP	Stop Clocks	GPTMCR
STOP	Stop Enable	QSMCR
STOP	Stop Control	RAMMCR
STOPP	Stop Prescaler	GPTMCR

Table D–8. Register Bit and Field Mnemonics (Continued)

STRB	Address Strobe/Data Strobe	CSOR[0:10], CSORBT
STSIM	Stop Mode System Integration Clock	SYNCR
STS[1:0]	Sample Time Selection	ADCTL0
SUPV	Supervisor/Unrestricted	ADCMCR
SUPV	Supervisor/Unrestricted	GPTMCR
SUPV	Supervisor/Unrestricted	QSMCR
SUPV	Supervisor/Unrestricted	SIMCR
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
TC	Transmit Complete	SCSR
TCIE	Transmit Complete Interrupt Enable	SCCR1
TDRE	Transmit Data Register Empty	SCSR
TE	Transmitter Enable	SCCR1
TIE	Transmit Interrupt Enable	SCCR1
TOF	Timer Overflow Flag	TFLG2
TOI	Timer Overflow Interrupt Enable	TMSK2
TR[0:F]	Transmit Data Ram	QSPI RAM
TST	Test Submodule Reset	RSR
TXD	Transmit Data	DDRQS
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	Frequency Control Bit (Prescale)	SYNCR
Y[5:0]	Frequency Control (Counter)	SYNCR
Z	Zero Flag	CCR
[CD:CA]	Channel Selection	ADCTL1

D

INDEX

– B –

– C –

– D –

– H –

– I –

– Q –

1ATX31096-0 Printed in USA 08/92 CUSTOM PRINTING CMTR6569 20,000 MCU YGACAA

MOTOROLA

Literature Distribution Centers:
USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.
EUROPE: Motorola Ltd.; European Literature Centre; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.
JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141, Japan.
ASIA PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Center, No. 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.

MC68HC16Z1UM/AD