MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

Order this document
by MC68HC16X1TS/D

# MC68HC16X1

# *Technical Summary*
# 16-Bit Modular Microcontroller

## 1 Introduction

The MC68HC16X1 microcontroller (MCU) is a high-speed 16-bit device that is upwardly code compatible with M68HC11 controllers. It is a member of the M68300/68HC16 Family of modular microcontrollers.

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

The MC68HC16X1 incorporates a true 16-bit CPU (CPU16), a single-chip integration module (SCIM), an 8/10-bit analog-to-digital converter (ADC), a queued serial module (QSM), a general-purpose timer (GPT), a 48-Kbyte masked ROM module (MRM), 2 Kbytes of standby RAM (SRAM), and 2 Kbytes of block-erasable flash EEPROM (BEFLASH). These modules are interconnected by the Motorola intermodule bus (IMB).

The MC68HC16X1 can either synthesize an internal clock signal from an external reference, or can use an external clock input directly. This MCU operates with a 32.768-kHz reference frequency.

High-density complementary metal-oxide semiconductor (HCMOS) architecture makes the basic power consumption of the MC68HC16X1 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.

**Ordering Information**

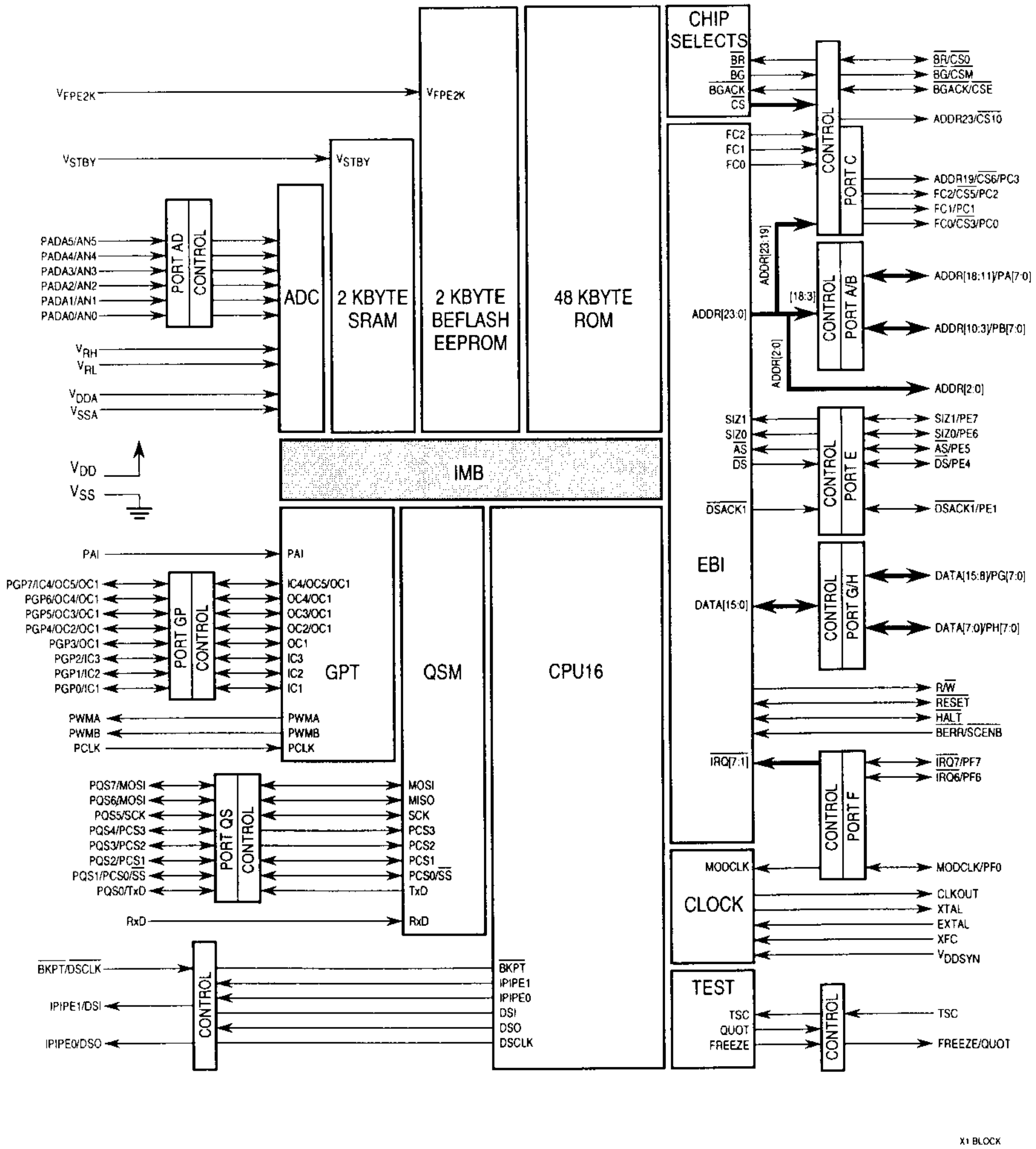
| Package Type | Frequency (MHz) | Temperature | Order Number |
|---|---|---|---|
| Quad Flat Pack | 16.78 | –40° to +85°C | MC68HC16X1CTH |

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

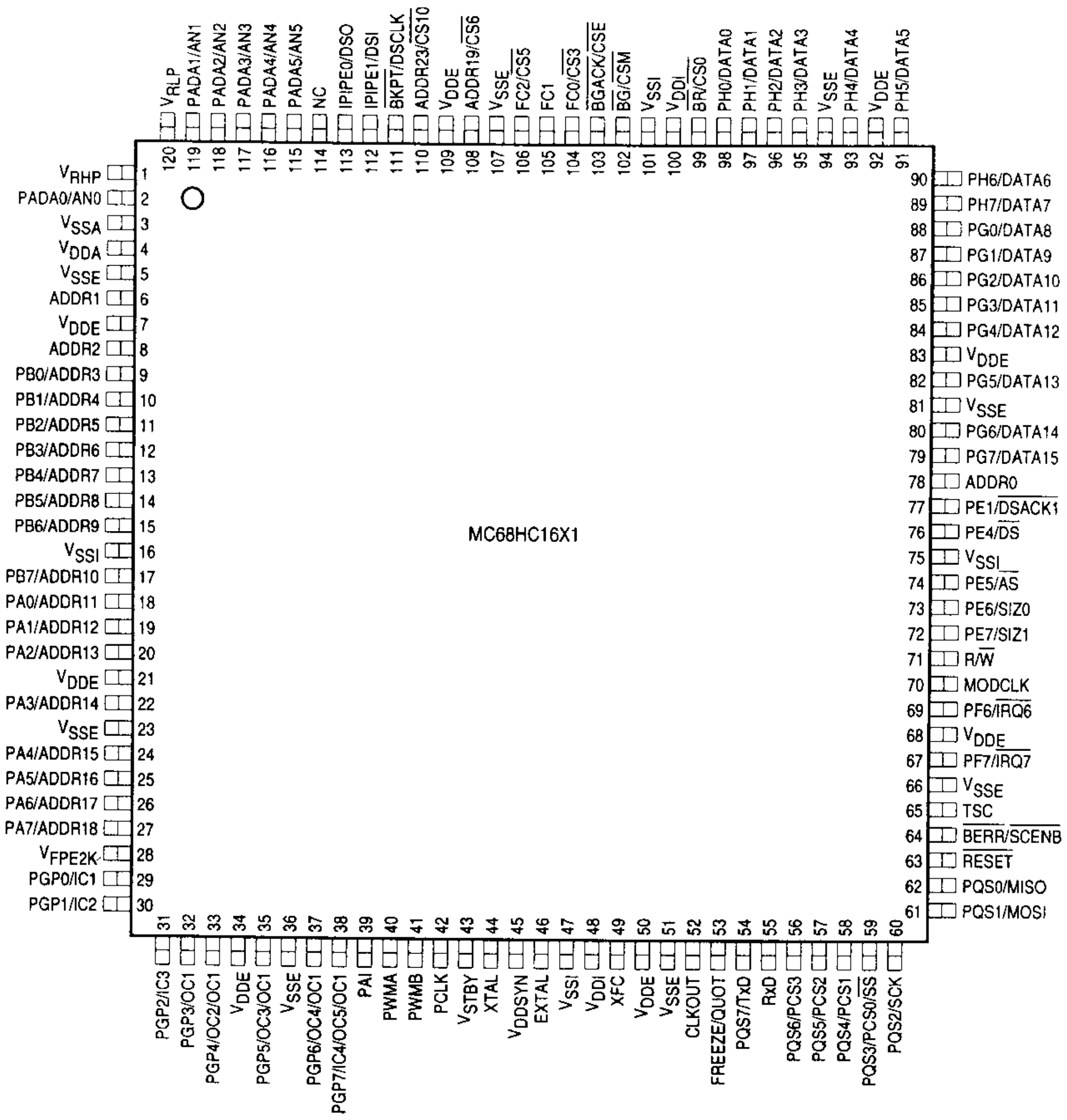
# Table of Contents

## 1.1 Features

- Central Processing Unit (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
- Single-Chip Integration Module (SCIM)
  - — Single-Chip or Expanded Modes of Operation
  - — External Bus Support in Expanded Mode
  - — Nine Programmable Chip Select Outputs
  - — System Protection Logic
  - — Watchdog Timer, Clock Monitor, and Bus Monitor
  - — Parallel Ports Option on Address and Data Bus in Single-Chip Mode
  - — Phase-Locked Loop (PLL) Clock System
- General-Purpose Timer (GPT)
  - — 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
  - — Optional External Clock Input
- 8/10-Bit Analog-to-Digital Converter (ADC)
  - — Six Channels, Eight Result Registers
  - — Eight Automated Modes
  - — Three Result Alignment Modes
- Queued Serial Module (QSM)
  - — Queued Serial Peripheral Interface (QSPI)
  - — Serial Communication Interface (SCI)
- 2 Kbyte Standby RAM Module (SRAM)
  - — External Standby Voltage Supply Input for Low-Power Standby Operation
- 48 Kbyte Masked ROM Module (MRM)
  - — Mask-Programmed ROM Registers Can Be Re-Mapped After Reset
  - — Boot ROM Capability
- 2 Kbyte Flash EEPROM with Independently Erasable Blocks (BEFLASH)
  - — Bulk/Block Erase and Byte/Word Programming with 12 Volt External Input



**MCU Block Diagram**



X1 120-PIN QFP

**120-Pin QFP Pinout**

## 1.2 Pin Description

The following table is a summary of MCU pin characteristics. All inputs detect CMOS logic levels. All outputs can be put in a high-impedance state, but the method of doing so differs depending upon pin function. Refer to the table of driver types for a description of output drivers. An entry in the **Discrete I/O** column of the table of pin characteristics indicates that a pin has an alternate I/O function. Port designation is given when it applies. Refer to the MCU block diagram for port organization.

**MCU Pin Characteristics**

| Pin Mnemonic | Output Driver | Input Synchronized | Input Hysteresis | Discrete I/O | Port Designation |
|---|---|---|---|---|---|
| ADDR23/CS10/ECLK | A | Y | N | — | — |
| ADDR19/CS6 | A | Y | — | O | C3 |
| ADDR[18:11] | A | Y | Y | I/O | A[7:0] |
| ADDR[10:3] | A | Y | Y | I/O | B[7:0] |
| ADDR[2:0] | A | Y | N | — | — |
| AN[5:0][1] | — | Y | Y | I | ADA[5:0] |
| AS | B | Y | Y | I/O | E5 |
| BERR | B | Y | N | — | — |
| BG/CSM | B | — | — | — | — |
| BGACK/CSE | B | Y | N | — | — |
| BKPT/DSCLK | — | Y | Y | — | — |
| BR/CS0 | B | Y | N | — | — |
| CLKOUT | A | — | — | — | — |
| DATA[15:8][1] | Aw | Y | Y | I/O | G[7:0] |
| DATA[7:0][1] | Aw | Y | Y | I/O | H[7:0] |
| DS | B | Y | Y | I/O | E4 |
| DSACK1 | B | Y | N | I/O | E1 |
| DSI/IPIPE1 | A | Y | Y | — | — |
| DSO/IPIPE0 | A | — | — | — | — |
| EXTAL[2] | — | — | — | — | — |
| FC2/CS5 | A | Y | — | O | C2 |
| FC1 | A | Y | — | O | C1 |
| FC0/CS3 | A | Y | — | O | C0 |
| FREEZE/QUOT | A | — | — | — | — |
| IC4/OC5 | A | Y | Y | I/O | GP7 |
| IC[3:1] | A | Y | Y | I/O | GP[2:0] |
| IRQ[7:6] | B | Y | Y | I/O | F[7:6] |

### MCU Pin Characteristics (Continued)

| Pin Mnemonic | Output Driver | Input Synchronized | Input Hysteresis | Discrete I/O | Port Designation |
|---|---|---|---|---|---|
| MISO[1] | Bo | Y | Y | I/O | QS6 |
| MODCLK[1] | B | Y | Y | I/O | F0 |
| MOSI[1] | Bo | Y | Y | I/O | QS7 |
| OC[4:1] | A | Y | Y | I/O | GP[6:3] |
| PAI[3] | — | Y | Y | I | — |
| PCLK[3] | — | Y | Y | I | — |
| PWMA, PWMB[4] | A | Y | Y | O | — |
| $R/\overline{W}$ | A | Y | N | — | — |
| $\overline{\text{RESET}}$ | Bo | Y | Y | — | — |
| RXD | — | Y | Y | I | — |
| SCK[1] | Bo | Y* | Y | I/O | QS5 |
| SIZ[1:0] | B | Y | N | I/O | E[7:6] |
| $\overline{\text{SS}}$ | Bo | Y | Y | I/O | QS1 |
| T2CLK | — | Y | Y | — | — |
| TPUCH[15:0] | A | Y | Y | — | — |
| TSC | — | Y | Y | — | — |
| TXD[1] | Bo | Y | Y | I/O | QS0 |
| $V_{RH}$[5] | — | — | — | — | — |
| $V_{RL}$[5] | — | — | — | — | — |
| XFC[2] | — | — | — | — | — |
| XTAL[2] | — | — | — | — | — |

NOTES

1. DATA[15:0] are synchronized during reset only. MODCLK, QSM 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.

### MCU Driver Types

| Type | I/O | Description |
|---|---|---|
| A | O | Output-only signals that are always driven. No external pull-up required. |
| Aw | O | Type A output with weak P-channel pull-up during reset. |
| B[1] | O | Three-state output that includes circuitry to pull up output before high impedance is established, to insure rapid rise time. An external holding resistor is required to maintain logic level while in the high-impedance state. |
| Bo | O | Type B output that can be operated in an open-drain mode. |

NOTES

1. Pins with this type of driver can only go into high-impedance state under certain conditions. The TSC signal can put all pins with this type of driver in high-impedance state.

**MCU Power Connections**

| | |
|---|---|
| $V_{DDA}/V_{SSA}$ | A/D Converter Power |
| $V_{DDSYN}$ | Clock Synthesizer Power |
| $V_{DDE}/V_{SSE}$ | External Peripheral Power (Source and Drain) |
| $V_{STBY}$ | Standby RAM Power/Clock Synthesizer Power |
| $V_{FPE}$ | Flash EEPROM Program/Erase Voltage |
| $V_{RHP}$ | ADC Reference Voltage High |
| $V_{RLP}$ | ADC Reference Voltage Low |

## 1.3 Signal Description

Use the following tables as a quick reference to MCU signal type and function.

**MCU Signal Characteristics**

| Signal Name | MCU Module | Signal Type | Active State |
|---|---|---|---|
| ADDR[23:0] | SCIM | Bus | — |
| AN[5:0] | ADC | Input | — |
| $\overline{\text{AS}}$ | SCIM | Output | 0 |
| $\overline{\text{BERR}}$ | SCIM | Input | 0 |
| $\overline{\text{BG}}$ | SCIM | Output | 0 |
| $\overline{\text{BGACK}}$ | SCIM | Input | 0 |
| $\overline{\text{BKPT}}$ | CPU16 | Input | 0 |
| $\overline{\text{BR}}$ | SCIM | Input | 0 |
| CLKOUT | SCIM | Output | — |
| $\overline{\text{CS10}}$, $\overline{\text{CS[6:5]}}$, $\overline{\text{CS3}}$, $\overline{\text{CS0}}$ | SCIM | Output | 0 |
| $\overline{\text{CSE}}$ | SCIM | Output | 0 |
| $\overline{\text{CSM}}$ | SCIM | Output | 0 |
| DATA[15:0] | SCIM | Bus | — |

## MCU Signal Characteristics (Continued)

| Signal Name | MCU Module | Signal Type | Active State |
|---|---|---|---|
| $\overline{\mathrm{DS}}$ | SCIM | Output | 0 |
| $\overline{\mathrm{DSACK1}}$ | SCIM | Input | 0 |
| DSCLK | CPU16 | Input | Serial Clock |
| DSI | CPU16 | Input | (Serial Data) |
| DSO | CPU16 | Output | (Serial Data) |
| ECLK | CPU16 | Output | — |
| EXTAL | SCIM | Input | — |
| FREEZE | SCIM | Output | 1 |
| IC[4:1] | GPT | Input | — |
| IPIPE0 | CPU16 | Output | — |
| IPIPE1 | CPU16 | Output | — |
| $\overline{\mathrm{IRQ[7:6]}}$ | SCIM | Input | 0 |
| MISO | QSM | Input/Output | — |
| MODCLK | SCIM | Input | — |
| MOSI | QSM | Input/Output | — |
| OC[5:1] | GPT | Output | — |
| PAI | GPT | Input | — |
| PCLK | GPT | Input | — |
| PCS[3:0] | QSM | Input/Output | — |
| PWMA, PWMB | GPT | Output | — |
| QUOT | SCIM | Output | — |
| $\mathrm{R}/\overline{\mathrm{W}}$ | SCIM | Output | — |
| $\overline{\mathrm{RESET}}$ | SCIM | Input/Output | 0 |
| RXD | QSM | Input | — |
| SCK | QSM | Input/Output | — |
| SIZ0/SIZ1 | SCIM | Output | — |
| $\overline{\mathrm{SS}}$ | QSM | Input | 0 |
| TSC | SCIM | Input | 1 |
| TXD | QSM | Input/Output | — |
| XFC | SCIM | Input | — |
| XTAL | SCIM | Output | — |

## MCU Signal Function

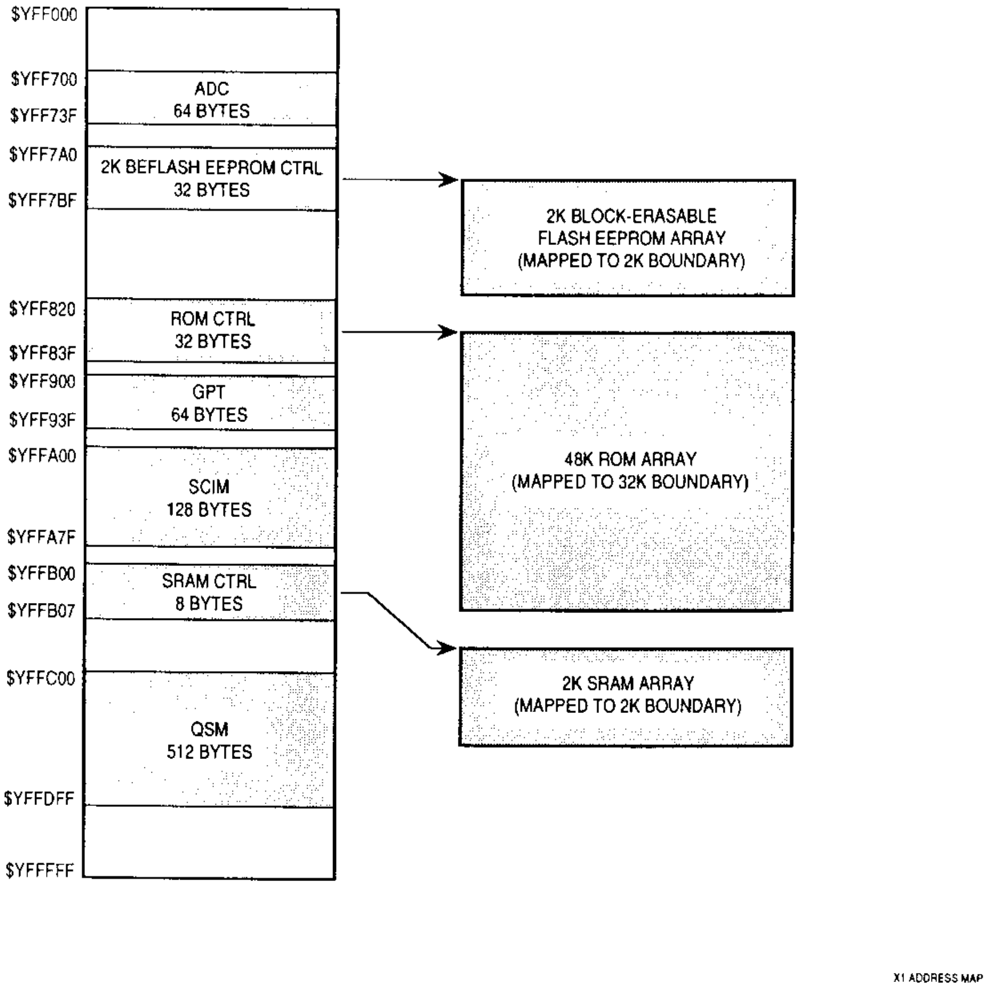
| Signal Name | Mnemonic | Function |
|---|---|---|
| Address Bus | ADDR[19:0] | 20-bit address bus used by CPU16 |
| Address Bus | ADDR[23:20] | 4 MSB on IMB, outputs follow ADDR19 |
| ADC Analog Input | AN[5:0] | Inputs to ADC MUX |
| Address Strobe | $\overline{\text{AS}}$ | Indicates that a valid address is on the address bus |
| Bus Grant | $\overline{\text{BG}}$ | Indicates that the MCU has relinquished the bus |
| Bus Grant Acknowledge | $\overline{\text{BGACK}}$ | Indicates that an external device has assumed bus mastership |
| Bus Error | $\overline{\text{BERR}}$ | Indicates that a bus error has occurred |
| Breakpoint | $\overline{\text{BKPT}}$ | Signals a hardware breakpoint to the CPU |
| Bus Request | $\overline{\text{BR}}$ | Indicates that an external device requires bus mastership |
| System Clockout | CLKOUT | System clock output |
| Emulation mode Chip Selects | $\overline{\text{CSE}}$, $\overline{\text{CSM}}$ | Select external emulation devices at internally-mapped addresses. Used to emulate I/O ports ($\overline{\text{CSE}}$) and memory ($\overline{\text{CSM}}$). |
| General-purpose Chip Selects | $\overline{\text{CS10}}$, $\overline{\text{CS[6:5]}}$, $\overline{\text{CS3}}$, $\overline{\text{CS0}}$ | Select external devices at programmed addresses |
| Data Bus | DATA[15:0] | 16-bit data bus |
| Data Strobe | $\overline{\text{DS}}$ | During a read cycle, indicates when it is possible for an external device to place data on the data bus. During a write cycle, indicates that valid data is on the data bus. |
| Data and Size Acknowledge | $\overline{\text{DSACK1}}$ | Asserted by external devices during asynchronous transfers to indicate receipt of data and width of receiving port. |
| Development Serial In, Out, Clock | DSI, DSO, DSCLK | Serial I/O and clock for background debug mode |
| External clock | ECLK | M6800 bus clock output. |
| 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 | IPIPE0<br>IPIPE1 | Indicate instruction pipeline activity |
| Interrupt Request | $\overline{\text{IRQ[7:1]}}$ | Request interrupt service from CPU16 |
| Master In Slave Out | MISO | Serial input to SPI in master mode;<br>serial output from SPI in slave mode |
| Clock Mode Select | MODCLK | Selects the source and type of system clock |
| Master Out Slave In | MOSI | Serial output from SPI in master mode;<br>serial input to SPI in slave mode |
| Quotient Out | QUOT | Provides the quotient bit of the polynomial divider |
| Read/Write | R/$\overline{\text{W}}$ | Indicates the direction of data transfer on the bus |
| Reset | $\overline{\text{RESET}}$ | System reset |
| SCI Receive Data | RXD | Serial input to SCI |
| SPI Serial Clock | SCK | Clock output from SPI in master mode;<br>clock input to SPI in slave mode |

## MCU Signal Function (Continued)

| Signal Name | Mnemonic | Function |
|---|---|---|
| Size | SIZ[1:0] | Indicate the size of an external bus transfer |
| Slave Select | $\overline{SS}$ | Selects SPI slave devices; assertion while a device is in master mode causes mode fault |
| Three-State Control | TSC | Places all output drivers in a high-impedance state |
| SCI Transmit Data | TXD | Serial output from SCI |
| External Filter Capacitor | XFC | Connection for external phase-locked loop filter capacitor |

## 1.4 Internal Register Address Map

Refer to the following internal address map of the MCU. Although there are 24 IMB address lines, the CPU16 uses only ADDR[19:0]. ADDR[23:20] are driven to the same logic state as ADDR19. Addresses $080000 to $F7FFFF are not accessible. The RAM array is positioned by the base address register in the RAM CTRL block. Reset disables the RAM array. Unimplemented blocks are mapped externally.



**MCU Address Map**

In the address map, Y = M111, where M is the modmap signal state on the IMB. M reflects the state of the modmap bit in the module configuration register of the single-chip integration module. In the M68HC16 microcontrollers, Y must equal $F. If M is cleared, IMB modules become inaccessible until a reset occurs. M can be written only once after reset.

### 1.5 Intermodule Bus

The IMB is a standardized bus developed to facilitate design of modular microcontrollers. It contains circuitry that supports exception processing, address space partitioning, multiple interrupt levels, and vectored interrupts. The standardized modules in the MCU communicate with one another and with external components via the IMB. Although the full IMB supports 24 address and 16 data lines, the MCU uses only 16 data lines and 20 address lines. Because the CPU16 uses only 20 address lines, ADDR[23:20] are tied to ADDR19 when processor driven. ADDR[23:20] are brought out to pins for test purposes.

## 2 Central Processing Unit

The CPU16 is a true 16-bit, high-speed device. It was designed to give M68HC11 users a path to higher performance while maintaining maximum compatibility with existing systems.

### 2.1 Overview

Ease of programming is an important consideration in using a microcontroller. The CPU16 instruction set is optimized for high performance. There are two 16-bit general-purpose accumulators and three 16-bit index registers. The CPU16 supports 8-bit (byte), 16-bit (word), and 32-bit (long-word) load and store operations, as well as 16- and 32-bit signed fractional operations. Program diagnosis is enhanced by a background debugging mode.

CPU16 memory space includes a 1 Mbyte data space and a 1 Mbyte program space. Twenty-bit addressing and transparent bank switching are used to implement extended memory. In addition, most instructions automatically handle bank boundaries.

The CPU16 includes instructions and hardware to implement control-oriented digital signal processing functions with a minimum of interfacing. A multiply and accumulate unit provides the capability to multiply signed 16-bit fractional numbers and store the resulting 32-bit fixed point product in a 36-bit accumulator. Modulo addressing supports finite impulse response filters.

Use of high-level languages is increasing as controller applications become more complex and control programs become larger. High-level languages aid rapid development of software, with less error, and are readily portable. The CPU16 instruction set supports high-level languages.

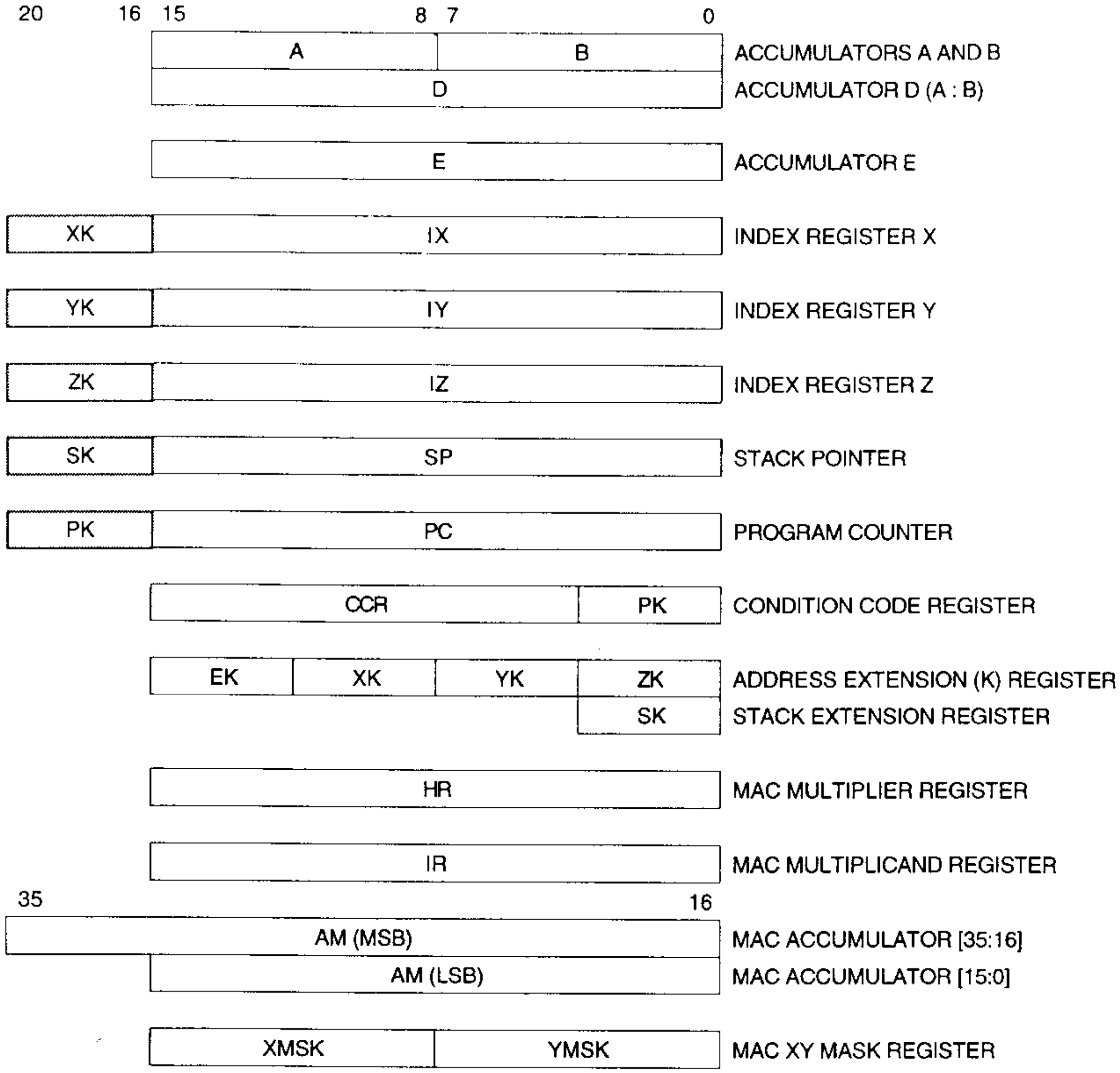
### 2.2 M68HC11 Compatibility

CPU16 architecture is a superset of M68HC11 architecture. All M68HC11 resources are available in the CPU16. M68HC11 instructions are either directly implemented in the CPU16, or have been replaced by instructions with an equivalent form. The instruction sets are source code compatible. Some instructions are executed differently in the CPU16. These instructions are mainly related to interrupt and exception processing. M68HC11 code that processes interrupts, handles stack frames, or manipulates the condition code register must be rewritten.

Execution times and number of cycles for all instructions are different, so that cycle-related delays and timed control routines may be affected.

The CPU16 also has several new or enhanced addressing modes. M68HC11 direct mode addressing has been replaced by a special form of indexed addressing that uses the new IZ register and a reset vector to provide greater flexibility.

## 2.3 Programmer's Model



Accumulator A — 8-bit general-purpose register
Accumulator B — 8-bit general-purpose register
Accumulator D — 16-bit register formed by concatenating accumulators A and B
Accumulator E — 16-bit general-purpose register
Index Register X — 16-bit indexing register, addressing extended by XK field in K register
Index Register Y — 16-bit indexing register, addressing extended by YK field in K register
Index Register Z — 16-bit indexing register, addressing extended by ZK field in K register
Stack pointer — 16-bit dedicated register, addressing extended by the SK register
Program Counter — 16-bit dedicated register, addressing extended by PK field in CCR
Condition Code Register — 16-bit register containing condition flags, interrupt priority mask, and the program counter address extension field
K Register — 16-bit register made up of four 4-bit address extension fields
SK Register — 4-bit register containing the stack pointer address extension field
H Register — 16-bit multiply and accumulate input (multiplier) register
I Register — 16-bit multiply and accumulate input (multiplicand) register
MAC Accumulator — 36-bit multiply and accumulate result register
XMSK, YMSK — Determine which bits change when an offset is added

**CCR** — Condition Code Register

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 5 | 4 | 3 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S | MV | H | EV | N | Z | V | C | INT | | SM | PK | |

The condition code register can be considered as two functional blocks. The MSB, which corresponds to the CCR in the M68HC11, contains the low-power stop control bit and processor status flags. The LSB contains the interrupt priority field, the DSP saturation mode control bit, and the program counter address extension field.

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 the accumulator M sign bit (AM35) has occurred.

H — Half Carry Flag
Set when a carry from bit 3 in accumulators A or B occurs during BCD addition.

EV — Extension Bit Overflow Flag
Set when an overflow into bit 31 of accumulator M 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 two's complement overflow occurs as the result of an operation.

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

INT[2:0] — Interrupt Priority Mask
The value of this field ($0 to $7) specifies the CPU16 interrupt priority level.

SM — Saturate Mode Bit
When SM is set, if either EV or MV is set, data read from accumulator M using TMRT 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 pseudolinear address.

## 2.4 Data Types

The CPU16 supports the following data types:

Bit data
8-bit (byte) and 16-bit (word) integers
32-bit long integers
16-bit and 32-bit signed fractions (MAC operations only)
20-bit effective address consisting of 16-bit page address plus 4-bit extension

A byte is 8 bits wide and can be accessed at any byte location. A word is composed of two consecutive bytes, and is addressed at the lower byte. Instruction fetches are always accessed on word boundaries. Word operands are normally accessed on word boundaries as well, but may be accessed on odd byte boundaries, with a substantial performance penalty.

To be compatible with the M68HC11, misaligned word transfers and misaligned stack accesses are allowed. Transferring a misaligned word requires two successive byte operations.

## 2.5 Addressing Modes

The CPU16 provides immediate, extended, indexed, inherent, accumulator offset, relative, and post-modified indexed addressing. Each addressing type encompasses one or more addressing modes. Six CPU16 addressing types are identical to M68HC11 addressing types. In addition, certain CPU16 features can be used to replace or extend M68HC11 direct addressing mode.

All modes generate ADDR[15:0]. This address is combined with ADDR[19:16] from an extension field to form a 20-bit effective address. Extension fields are part of a bank switching scheme that provides the CPU16 with a 1 Mbyte address space. Bank switching is transparent to most instructions. ADDR[19:16] of the effective address change when an access crosses a bank boundary. However, it is important to note that the value of the associated extension field is dependent on the type of instruction, and generally does not change when this occurs.

In the immediate modes, the instruction argument is contained in bytes or words immediately following the instruction. The effective address is the address of the byte following the instruction. The AIS, AIX/Y/Z, ADDD and ADDE instructions have an extended 8-bit mode where the immediate value is an 8-bit signed number that is sign-extended to 16 bits, then added to the appropriate register — this decreases execution time.

Extended mode instructions contain ADDR[15:0] in the word following the opcode. The effective address is formed by concatenating EK and the 16-bit extension.

In the indexed modes, registers IX, IY, and IZ, together with their associated extension fields, are used to calculate the effective address. Signed 16-bit mode and signed 20-bit mode are extensions to the M68HC11 indexed addressing mode.

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

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

For 20-bit mode, a 20-bit signed offset is added to the value contained in the index register. This mode is used for JMP and JSR instructions.

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

Accumulator offset mode adds the contents of 16-bit accumulator E to one of the index registers and its associated extension field to form the effective address. This mode allows use of index registers and an accumulator within loops without corrupting accumulator D.

Relative modes are used for branch and long branch instructions. A byte or word signed two's complement offset is added to the program counter if the branch condition is satisfied. The new PC value, concatenated with the PK field, is the effective address.

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

In M68HC11 systems, direct mode can be used to perform rapid accesses to RAM or I/O mapped into page 0 ($0000 to $00FF), but the CPU16 uses the first 512 bytes of page 0 for exception vectors. To compensate, the ZK field and index register Z have been assigned reset initialization vectors. By resetting the ZK field to a chosen page and using 8-bit unsigned index mode with IZ, a programmer can access useful data structures anywhere in the address map.

## 2.6 Instruction Set

The CPU16 has an 8-bit instruction set. It uses a prebyte to support a multipage opcode map. This arrangement makes it possible to fetch an 8-bit operand simultaneously with a page 0 opcode. If a program makes maximum use of 8-bit offset indexed addressing mode, it will have a significantly smaller instruction space.

The instruction set is based upon that of the M68HC11, but the opcode map has been rearranged to maximize performance with a 16-bit data bus. All M68HC11 instructions are supported by the CPU16, although they can be executed differently. Most M68HC11 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.

The CPU16 has a full range of 16-bit arithmetic and logic instructions, including signed and unsigned multiplication and division. New instructions have been added to support extended addressing and digital signal processing.

The following table is a summary of the CPU16 instruction set. Because it is only affected by a few instructions, the LSB of the condition code register is not shown in the table. Instructions that affect the interrupt mask and PK field are noted.

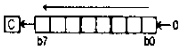
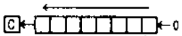
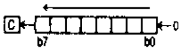
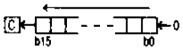
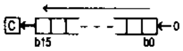
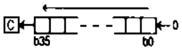
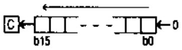
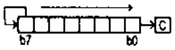
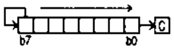
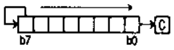
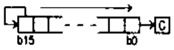
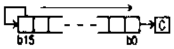
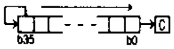
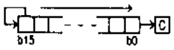
## Instruction Set Summary

| Mnemonic | Operation | Description | Address | Instruction | | | Condition Codes | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Mode | 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 | Instruction | | | Condition Codes | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Mode | Opcode | Operand | Cycles | S | MV | H | EV | N | Z | V | C |
| ADDB | Add to B | (B) + (M) ⇒ B | IND8, X | C1 | ff | 6 | — | — | Δ | — | Δ | Δ | Δ | Δ |
| | | | IND8, Y | D1 | ff | 6 | | | | | | | | |
| | | | IND8, Z | E1 | ff | 6 | | | | | | | | |
| | | | IMM8 | F1 | ii | 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 | ff | 6 | — | — | — | — | Δ | Δ | Δ | Δ |
| | | | IND8, Y | 91 | ff | 6 | | | | | | | | |
| | | | IND8, Z | A1 | ff | 6 | | | | | | | | |
| | | | IMM8 | FC | ii | 2 | | | | | | | | |
| | | | E, X | 2781 | — | 6 | | | | | | | | |
| | | | E, Y | 2791 | — | 6 | | | | | | | | |
| | | | E, Z | 27A1 | — | 6 | | | | | | | | |
| | | | IMM16 | 37B1 | jjkk | 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 | | | | | | | | |

## Instruction Set Summary (Continued)

| Mnemonic | Operation | Description | Address | Instruction | | | Condition Codes | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Mode | Opcode | Operand | Cycles | 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 | — | 4 | — | — | — | 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[4] | Branch if Carry Clear | If C = 0, branch | REL8 | B4 | rr | 6, 2 | — | — | — | — | — | — | — | — |

## Instruction Set Summary (Continued)

| Mnemonic | Operation | Description | Address | Instruction | | | Condition Codes | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Mode | Opcode | Operand | Cycles | S | MV | H | EV | N | Z | V | C |
| BCLR | Clear Bit(s) | (M) • $\overline{\text{(Mask)}}$ ⇒ M | IND16, X<br>IND16, Y<br>IND16, Z<br>EXT<br>IND8, X<br>IND8, Y<br>IND8, Z | 08<br>18<br>28<br>38<br>1708<br>1718<br>1728 | mm gggg<br>mm gggg<br>mm gggg<br>mm hh ll<br>mm ff<br>mm ff<br>mm ff | 8<br>8<br>8<br>8<br>8<br>8<br>8 | — | — | — | — | Δ | Δ | 0 | — |
| BCLRW | Clear Bit(s) Word | (M : M + 1) • $\overline{\text{(Mask)}}$ ⇒ M : M + 1 | IND16, X<br>IND16, Y<br>IND16, Z<br>EXT | 2708<br>2718<br>2728<br>2738 | gggg mmmm<br>gggg mmmm<br>gggg mmmm<br>hh ll mmmm | 10<br>10<br>10<br>10 | — | — | — | — | Δ | Δ | 0 | — |
| BCS[4] | Branch if Carry Set | If C = 1, branch | REL8 | B5 | rr | 6, 2 | — | — | — | — | — | — | — | — |
| BEQ[4] | Branch if Equal | If Z = 1, branch | REL8 | B7 | rr | 6, 2 | — | — | — | — | — | — | — | — |
| BGE[4] | 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 [4] | Branch if Greater Than Zero | If Z + (N ⊕ V) = 0, branch | REL8 | BE | rr | 6, 2 | — | — | — | — | — | — | — | — |
| BHI [4] | Branch if Higher | If C + Z = 0, branch | REL8 | B2 | rr | 6, 2 | — | — | — | — | — | — | — | — |
| BITA | Bit Test A | (A) • (M) | IND8, X<br>IND8, Y<br>IND8, Z<br>IMM8<br>IND16, X<br>IND16, Y<br>IND16, Z<br>EXT<br>E, X<br>E, Y<br>E, Z | 49<br>59<br>69<br>79<br>1749<br>1759<br>1769<br>1779<br>2749<br>2759<br>2769 | ff<br>ff<br>ff<br>ii<br>gggg<br>gggg<br>gggg<br>hh ll<br>—<br>—<br>— | 6<br>6<br>6<br>2<br>6<br>6<br>6<br>6<br>6<br>6<br>6 | — | — | — | — | Δ | Δ | 0 | — |
| BITB | Bit Test B | (B) • (M) | IND8, X<br>IND8, Y<br>IND8, Z<br>IMM8<br>IND16, X<br>IND16, Y<br>IND16, Z<br>EXT<br>E, X<br>E, Y<br>E, Z | C9<br>D9<br>E9<br>F9<br>17C9<br>17D9<br>17E9<br>17F9<br>27C9<br>27D9<br>27E9 | ff<br>ff<br>ff<br>ii<br>gggg<br>gggg<br>gggg<br>hh ll<br>—<br>—<br>— | 6<br>6<br>6<br>2<br>6<br>6<br>6<br>6<br>6<br>6<br>6 | — | — | — | — | Δ | Δ | 0 | — |
| BLE [4] | Branch if Less Than or Equal to Zero | If Z + (N ⊕ V) = 1, branch | REL8 | BF | rr | 6, 2 | — | — | — | — | — | — | — | — |
| BLS[4] | Branch if Lower or Same | If C + Z = 1, branch | REL8 | B3 | rr | 6, 2 | — | — | — | — | — | — | — | — |
| BLT[4] | Branch if Less Than Zero | If N ⊕ V = 1, branch | REL8 | BD | rr | 6, 2 | — | — | — | — | — | — | — | — |
| BMI [4] | Branch if Minus | If N = 1, branch | REL8 | BB | rr | 6, 2 | — | — | — | — | — | — | — | — |
| BNE [4] | Branch if Not Equal | If Z = 0, branch | REL8 | B6 | rr | 6, 2 | — | — | — | — | — | — | — | — |
| BPL[4] | Branch if Plus | If N = 0, branch | REL8 | BA | rr | 6, 2 | — | — | — | — | — | — | — | — |
| BRA | Branch Always | If 1 = 1, branch | REL8 | B0 | rr | 6 | — | — | — | — | — | — | — | — |
| BRCLR[4] | Branch if Bit(s) Clear | If (M) • (Mask) = 0, branch | IND8, X<br>IND8, Y<br>IND8, Z<br>IND16, X<br>IND16, Y<br>IND16, Z<br>EXT | CB<br>DB<br>EB<br>0A<br>1A<br>2A<br>3A | mm ff rr<br>mm ff rr<br>mm ff rr<br>mm gggg rrrr<br>mm gggg rrrr<br>mm gggg rrrr<br>mm hh ll rrrr | 10, 12<br>10, 12<br>10, 12<br>10, 14<br>10, 14<br>10, 14<br>10, 14 | — | — | — | — | — | — | — | — |
| BRN | Branch Never | If 1 = 0, branch | REL8 | B1 | rr | 2 | — | — | — | — | — | — | — | — |

## Instruction Set Summary (Continued)

| Mnemonic | Operation | Description | Address | Instruction | | | Condition Codes | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Mode | Opcode | Operand | Cycles | S | MV | H | EV | N | Z | V | C |
| BRSET[4] | Branch if Bit(s) Set | If ($\overline{M}$) • (Mask) = 0, branch | IND8, X | 8B | mm ff rr | 10, 12 | — | — | — | — | — | — | — | — |
| | | | IND8, Y | 9B | mm ff rr | 10, 12 | | | | | | | | |
| | | | IND8, Z | AB | mm ff rr | 10, 12 | | | | | | | | |
| | | | IND16, X | 0B | mm gggg rrr | 10, 14 | | | | | | | | |
| | | | IND16, Y | 1B | mm gggg rrr | 10, 14 | | | | | | | | |
| | | | IND16, Z | 2B | mm gggg rrr | 10, 14 | | | | | | | | |
| | | | EXT | 3B | mm hh ll rrr | 10, 14 | | | | | | | | |
| 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 | gggg mmmm | 10 | — | — | — | — | Δ | Δ | 0 | — |
| | | | IND16, Y | 2719 | gggg mmmm | 10 | | | | | | | | |
| | | | IND16, Z | 2729 | gggg mmmm | 10 | | | | | | | | |
| | | | EXT | 2739 | hh ll mmmm | 10 | | | | | | | | |
| BSR | Branch to Subroutine | (PK : PC) – 2 ⇒ PK : PC<br>Push (PC)<br>(SK : SP) – 2 ⇒ SK : SP<br>Push (CCR)<br>(SK : SP) – 2 ⇒ SK : SP<br>(PK:PC) + Offset ⇒ PK:PC | REL8 | 36 | rr | 10 | — | — | — | — | — | — | — | — |
| BVC[4] | Branch if Overflow Clear | If V = 0, branch | REL8 | B8 | rr | 6, 2 | — | — | — | — | — | — | — | — |
| BVS[4] | Branch if Overflow Set | If V = 1, branch | REL8 | B9 | rr | 6, 2 | — | — | — | — | — | — | — | — |
| CBA | Compare A to B | (A) – (B) | INH | 371B | — | 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 | Instruction | | | Condition Codes | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Mode | Opcode | Operand | Cycles | 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 | jjkk | 4 | — | — | — | — | Δ | Δ | Δ | Δ |
| | | | IND16, X | 3748 | gggg | 6 | | | | | | | | |
| | | | IND16, Y | 3758 | gggg | 6 | | | | | | | | |
| | | | IND16, Z | 3768 | gggg | 6 | | | | | | | | |
| | | | EXT | 3778 | hhll | 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 | Instruction | | | Condition Codes | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Mode | Opcode | Operand | Cycles | S | MV | H | EV | N | Z | V | C |
| CPZ | Compare IZ to Memory | (IZ) – (M : M + 1) | IND8, X<br>IND8, Y<br>IND8, Z<br>IND16, X<br>IND16, Y<br>IND16, Z<br>EXT<br>IMM16 | 4E<br>5E<br>6E<br>174E<br>175E<br>176E<br>177E<br>377E | ff<br>ff<br>ff<br>gggg<br>gggg<br>gggg<br>hh ll<br>jj kk | 6<br>6<br>6<br>6<br>6<br>6<br>6<br>4 | — | — | — | — | Δ | Δ | Δ | Δ |
| DAA | Decimal Adjust A | $(A)_{10}$ | INH | 3721 | — | 2 | — | — | — | — | Δ | Δ | U | Δ |
| DEC | Decrement Memory | (M) – $01 ⇒ M | IND8, X<br>IND8, Y<br>IND8, Z<br>IND16, X<br>IND16, Y<br>IND16, Z<br>EXT | 01<br>11<br>21<br>1701<br>1711<br>1721<br>1731 | ff<br>ff<br>ff<br>gggg<br>gggg<br>gggg<br>hh ll | 8<br>8<br>8<br>8<br>8<br>8<br>8 | — | — | — | — | Δ | Δ | Δ | — |
| 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<br>IND16, Y<br>IND16, Z<br>EXT | 2701<br>2711<br>2721<br>2731 | gggg<br>gggg<br>gggg<br>hh ll | 8<br>8<br>8<br>8 | — | — | — | — | Δ | Δ | Δ | — |
| EDIV | Extended Unsigned Divide | (E : D) / (IX)<br>Quotient ⇒ IX<br>Remainder ⇒ D | INH | 3728 | — | 24 | — | — | — | — | Δ | Δ | Δ | Δ |
| EDIVS | Extended Signed Divide | (E : D) / (IX)<br>Quotient ⇒ IX<br>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<br>IND8, Y<br>IND8, Z<br>IMM8<br>IND16, X<br>IND16, Y<br>IND16, Z<br>EXT<br>E, X<br>E, Y<br>E, Z | 44<br>54<br>64<br>74<br>1744<br>1754<br>1764<br>1774<br>2744<br>2754<br>2764 | ff<br>ff<br>ff<br>ii<br>gggg<br>gggg<br>gggg<br>hh ll<br>—<br>—<br>— | 6<br>6<br>6<br>2<br>6<br>6<br>6<br>6<br>6<br>6<br>6 | — | — | — | — | Δ | Δ | 0 | — |
| EORB | Exclusive OR B | (B) ⊕ (M) ⇒ B | IND8, X<br>IND8, Y<br>IND8, Z<br>IMM8<br>IND16, X<br>IND16, Y<br>IND16, Z<br>EXT<br>E, X<br>E, Y<br>E, Z | C4<br>D4<br>E4<br>F4<br>17C4<br>17D4<br>17E4<br>17F4<br>27C4<br>27D4<br>27E4 | ff<br>ff<br>ff<br>ii<br>gggg<br>gggg<br>gggg<br>hh ll<br>—<br>—<br>— | 6<br>6<br>6<br>2<br>6<br>6<br>6<br>6<br>6<br>6<br>6 | — | — | — | — | Δ | Δ | 0 | — |
| EORD | Exclusive OR D | (D) ⊕ (M : M + 1) ⇒ D | IND8, X<br>IND8, Y<br>IND8, Z<br>E, X<br>E, Y<br>E, Z<br>IMM16<br>IND16, X<br>IND16, Y<br>IND16, Z<br>EXT | 84<br>94<br>A4<br>2784<br>2794<br>27A4<br>37B4<br>37C4<br>37D4<br>37E4<br>37F4 | ff<br>ff<br>ff<br>—<br>—<br>—<br>jjkk<br>gggg<br>gggg<br>gggg<br>hhll | 6<br>6<br>6<br>6<br>6<br>6<br>4<br>6<br>6<br>6<br>6 | — | — | — | — | Δ | Δ | 0 | — |
| EORE | Exclusive OR E | (E) ⊕ (M : M + 1) ⇒ E | IMM16<br>IND16, X<br>IND16, Y<br>IND16, Z<br>EXT | 3734<br>3744<br>3754<br>3764<br>3774 | jj kk<br>gggg<br>gggg<br>gggg<br>hh ll | 4<br>6<br>6<br>6<br>6 | — | — | — | — | Δ | Δ | 0 | — |

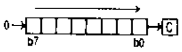
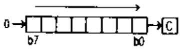
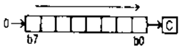
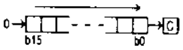
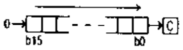
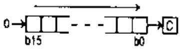
## Instruction Set Summary (Continued)

| Mnemonic | Operation | Description | Address | Instruction | | | Condition Codes | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Mode | Opcode | Operand | Cycles | S | MV | H | EV | N | Z | V | C |
| FDIV | Fractional Unsigned Divide | (D) / (IX) ⇒ IX<br>Remainder ⇒ D | INH | 372B | — | 22 | — | — | — | — | — | Δ | Δ | Δ |
| FMULS | Fractional Signed Multiply | (E) * (D) ⇒ E : D[31:1]<br>0 ⇒ D[0] | INH | 3727 | — | 8 | — | — | — | — | Δ | Δ | Δ | Δ |
| IDIV | Integer Divide | (D) / (IX) ⇒ IX;<br>Remainder ⇒ D | INH | 372A | | 22 | — | — | — | — | — | Δ | 0 | Δ |
| INC | Increment Memory | (M) + $01 ⇒ M | IND8, X<br>IND8, Y<br>IND8, Z<br>IND16, X<br>IND16, Y<br>IND16, Z<br>EXT | 03<br>13<br>23<br>1703<br>1713<br>1723<br>1733 | ff<br>ff<br>ff<br>gggg<br>gggg<br>gggg<br>hh ll | 8<br>8<br>8<br>8<br>8<br>8<br>8 | — | — | — | — | 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<br>⇒ M : M + 1 | IND16, X<br>IND16, Y<br>IND16, Z<br>EXT | 2703<br>2713<br>2723<br>2733 | gggg<br>gggg<br>gggg<br>hh ll | 8<br>8<br>8<br>8 | — | — | — | — | Δ | Δ | Δ | — |
| JMP | Jump | (ea) ⇒ PK : PC | IND20, X<br>IND20, Y<br>IND20, Z<br>EXT20 | 4B<br>5B<br>6B<br>7A | zg gggg<br>zg gggg<br>zg gggg<br>zb hh ll | 8<br>8<br>8<br>6 | — | — | — | — | — | — | — | — |
| JSR | Jump to Subroutine | Push (PC)<br>(SK : SP) − 2 ⇒ SK : SP<br>Push (CCR)<br>(SK : SP) − 2 ⇒ SK : SP<br>(ea) ⇒ PK : PC | IND20, X<br>IND20, Y<br>IND20, Z<br>EXT20 | 89<br>99<br>A9<br>FA | zg gggg<br>zg gggg<br>zg gggg<br>zb hh ll | 12<br>12<br>12<br>10 | — | — | — | — | — | — | — | — |
| LBCC[4] | Long Branch if Carry Clear | If C = 0, branch | REL16 | 3784 | rrrr | 6, 4 | — | — | — | — | — | — | — | — |
| LBCS[4] | Long Branch if Carry Set | If C = 1, branch | REL16 | 3785 | rrrr | 6, 4 | — | — | — | — | — | — | — | — |
| LBEQ[4] | Long Branch if Equal | If Z = 1, branch | REL16 | 3787 | rrrr | 6, 4 | — | — | — | — | — | — | — | — |
| LBEV[4] | Long Branch if EV Set | If EV = 1, branch | REL16 | 3791 | rrrr | 6, 4 | — | — | — | — | — | — | — | — |
| LBGE[4] | Long Branch if Greater Than or Equal to Zero | If N ⊕ V = 0, branch | REL16 | 378C | rrrr | 6, 4 | — | — | — | — | — | — | — | — |
| LBGT[4] | Long Branch if Greater Than Zero | If Z + (N ⊕ V) = 0, branch | REL16 | 378E | rrrr | 6, 4 | — | — | — | — | — | — | — | — |
| LBHI[4] | Long Branch if Higher | If C + Z = 0, branch | REL16 | 3782 | rrrr | 6, 4 | — | — | — | — | — | — | — | — |
| LBLE[4] | Long Branch if Less Than or Equal to Zero | If Z + (N ⊕ V) = 1, branch | REL16 | 378F | rrrr | 6, 4 | — | — | — | — | — | — | — | — |
| LBLS[4] | Long Branch if Lower or Same | If C + Z = 1, branch | REL16 | 3783 | rrrr | 6, 4 | — | — | — | — | — | — | — | — |
| LBLT[4] | Long Branch if Less Than Zero | If N ⊕ V = 1, branch | REL16 | 378D | rrrr | 6, 4 | — | — | — | — | — | — | — | — |
| LBMI[4] | Long Branch if Minus | If N = 1, branch | REL16 | 378B | rrrr | 6, 4 | — | — | — | — | — | — | — | — |
| LBMV[4] | Long Branch if MV Set | If MV = 1, branch | REL16 | 3790 | rrrr | 6, 4 | | | | | | | | |
| LBNE[4] | Long Branch if Not Equal | If Z = 0, branch | REL16 | 3786 | rrrr | 6, 4 | — | — | — | — | — | — | — | — |
| LBPL[4] | 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)<br>(SK : SP) − 2 ⇒ SK : SP<br>Push (CCR)<br>(SK : SP) − 2 ⇒ SK : SP<br>(PK : PC) + Offset ⇒<br>PK : PC | REL16 | 27F9 | rrrr | 10 | — | — | — | — | — | — | — | — |
| LBVC[4] | Long Branch if Overflow Clear | If V = 0, branch | REL16 | 3788 | rrrr | 6, 4 | — | — | — | — | — | — | — | — |
| LBVS[4] | Long Branch if Overflow Set | If V = 1, branch | REL16 | 3789 | rrrr | 6, 4 | — | — | — | — | — | — | — | — |

## Instruction Set Summary (Continued)

| Mnemonic | Operation | Description | Address | Instruction | | | Condition Codes | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Mode | 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<br>(M + 2 : M + 3) ⇒ D | EXT | 2771 | hh ll | 8 | — | — | — | — | — | — | — | — |
| LDHI | Initialize H and I | $(M : M + 1)_X \Rightarrow H\,R$<br>$(M : M + 1)_Y \Rightarrow I\,R$ | EXT | 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 | | | | | | | | |

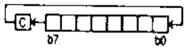
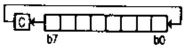
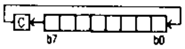
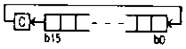
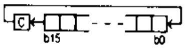
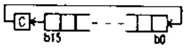
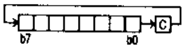
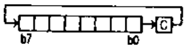
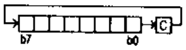
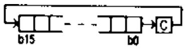
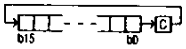
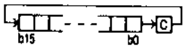
## Instruction Set Summary (Continued)

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

## Instruction Set Summary (Continued)

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

## Instruction Set Summary (Continued)

| Mnemonic | Operation | Description | Address Mode | Instruction |  |  | Condition Codes |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  |  |  | Opcode | Operand | Cycles | S | MV | H | EV | N | Z | V | C |
| ROL | Rotate Left |  | IND8, X<br>IND8, Y<br>IND8, Z<br>IND16, X<br>IND16, Y<br>IND16, Z<br>EXT | 0C<br>1C<br>2C<br>170C<br>171C<br>172C<br>173C | ff<br>ff<br>ff<br>gggg<br>gggg<br>gggg<br>hh ll | 8<br>8<br>8<br>8<br>8<br>8<br>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<br>IND16, Y<br>IND16, Z<br>EXT | 270C<br>271C<br>272C<br>273C | gggg<br>gggg<br>gggg<br>hh ll | 8<br>8<br>8<br>8 | — | — | — | — | Δ | Δ | Δ | Δ |
| ROR | Rotate Right |  | IND8, X<br>IND8, Y<br>IND8, Z<br>IND16, X<br>IND16, Y<br>IND16, Z<br>EXT | 0E<br>1E<br>2E<br>170E<br>171E<br>172E<br>173E | ff<br>ff<br>ff<br>gggg<br>gggg<br>gggg<br>hh ll | 8<br>8<br>8<br>8<br>8<br>8<br>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<br>IND16, Y<br>IND16, Z<br>EXT | 270E<br>271E<br>272E<br>273E | gggg<br>gggg<br>gggg<br>hh ll | 8<br>8<br>8<br>8 | — | — | — | — | Δ | Δ | Δ | Δ |
| RTI[2] | Return from Interrupt | (SK : SP) + 2 ⇒ SK : SP<br>Pull CCR<br>(SK : SP) + 2 ⇒ SK : SP<br>Pull PC<br>(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<br>Pull PK<br>(SK : SP) + 2 ⇒ SK : SP<br>Pull PC<br>(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<br>IND8, Y<br>IND8, Z<br>IMM8<br>IND16, X<br>IND16, Y<br>IND16, Z<br>EXT<br>E, X<br>E, Y<br>E, Z | 42<br>52<br>62<br>72<br>1742<br>1752<br>1762<br>1772<br>2742<br>2752<br>2762 | ff<br>ff<br>ff<br>ii<br>gggg<br>gggg<br>gggg<br>hh ll<br>—<br>—<br>— | 6<br>6<br>6<br>2<br>6<br>6<br>6<br>6<br>6<br>6<br>6 | — | — | — | — | Δ | Δ | Δ | Δ |

## Instruction Set Summary (Continued)

| Mnemonic | Operation | Description | Address | Instruction | | | Condition Codes | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Mode | Opcode | Operand | Cycles | 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 | jj kk | 4 | | | | | | | | |
| | | | IND16, X | 37C2 | gggg | 6 | | | | | | | | |
| | | | IND16, Y | 37D2 | gggg | 6 | | | | | | | | |
| | | | IND16, Z | 37E2 | gggg | 6 | | | | | | | | |
| | | | EXT | 37F2 | hh ll | 6 | | | | | | | | |
| SBCE | Subtract with Carry from E | (E) – (M : M + 1) – C ⇒ E | IMM16 | 3732 | jj kk | 4 | — | — | — | — | Δ | Δ | Δ | Δ |
| | | | IND16, X | 3742 | gggg | 6 | | | | | | | | |
| | | | IND16, Y | 3752 | gggg | 6 | | | | | | | | |
| | | | IND16, Z | 3762 | gggg | 6 | | | | | | | | |
| | | | EXT | 3772 | hh ll | 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 | 6 | — | — | — | — | Δ | Δ | 0 | — |
| | | | IND8, Y | 9A | ff | 6 | | | | | | | | |
| | | | IND8, Z | AA | ff | 6 | | | | | | | | |
| | | | 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 | hh ll | 6 | | | | | | | | |
| STE | Store E | (E) ⇒ M : M + 1 | IND16, X | 374A | gggg | 6 | — | — | — | — | Δ | Δ | 0 | — |
| | | | IND16, Y | 375A | gggg | 6 | | | | | | | | |
| | | | IND16, Z | 376A | gggg | 6 | | | | | | | | |
| | | | EXT | 377A | hh ll | 6 | | | | | | | | |
| STED | Store Concatenated D and E | (E) ⇒ M : M + 1<br>(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 | Instruction | | | Condition Codes | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Mode | Opcode | Operand | Cycles | S | MV | H | EV | N | Z | V | C |
| STX | Store IX | (IX) ⇒ M : M + 1 | IND8, X<br>IND8, Y<br>IND8, Z<br>IND16, X<br>IND16, Y<br>IND16, Z<br>EXT | 8C<br>9C<br>AC<br>178C<br>179C<br>17AC<br>17BC | ff<br>ff<br>ff<br>gggg<br>gggg<br>gggg<br>hh ll | 4<br>4<br>4<br>6<br>6<br>6<br>6 | — | — | — | — | Δ | Δ | 0 | — |
| STY | Store IY | (IY) ⇒ M : M + 1 | IND8, X<br>IND8, Y<br>IND8, Z<br>IND16, X<br>IND16, Y<br>IND16, Z<br>EXT | 8D<br>9D<br>AD<br>178D<br>179D<br>17AD<br>17BD | ff<br>ff<br>ff<br>gggg<br>gggg<br>gggg<br>hh ll | 4<br>4<br>4<br>6<br>6<br>6<br>6 | — | — | — | — | Δ | Δ | 0 | — |
| STZ | Store Z | (IZ) ⇒ M : M + 1 | IND8, X<br>IND8, Y<br>IND8, Z<br>IND16, X<br>IND16, Y<br>IND16, Z<br>EXT | 8E<br>9E<br>AE<br>178E<br>179E<br>17AE<br>17BE | ff<br>ff<br>ff<br>gggg<br>gggg<br>gggg<br>hh ll | 4<br>4<br>4<br>6<br>6<br>6<br>6 | — | — | — | — | Δ | Δ | 0 | — |
| SUBA | Subtract from A | (A) – (M) ⇒ A | IND8, X<br>IND8, Y<br>IND8, Z<br>IMM8<br>IND16, X<br>IND16, Y<br>IND16, Z<br>EXT<br>E, X<br>E, Y<br>E, Z | 40<br>50<br>60<br>70<br>1740<br>1750<br>1760<br>1770<br>2740<br>2750<br>2760 | ff<br>ff<br>ff<br>ii<br>gggg<br>gggg<br>gggg<br>hh ll<br>—<br>—<br>— | 6<br>6<br>6<br>2<br>6<br>6<br>6<br>6<br>6<br>6<br>6 | — | — | — | — | Δ | Δ | Δ | Δ |
| SUBB | Subtract from B | (B) – (M) ⇒ B | IND8, X<br>IND8, Y<br>IND8, Z<br>IMM8<br>IND16, X<br>IND16, Y<br>IND16, Z<br>EXT<br>E, X<br>E, Y<br>E, Z | C0<br>D0<br>E0<br>F0<br>17C0<br>17D0<br>17E0<br>17F0<br>27C0<br>27D0<br>27E0 | ff<br>ff<br>ff<br>ii<br>gggg<br>gggg<br>gggg<br>hh ll<br>—<br>—<br>— | 6<br>6<br>6<br>2<br>6<br>6<br>6<br>6<br>6<br>6<br>6 | — | — | — | — | Δ | Δ | Δ | Δ |
| SUBD | Subtract from D | (D) – (M : M + 1) ⇒ D | IND8, X<br>IND8, Y<br>IND8, Z<br>E, X<br>E, Y<br>E, Z<br>IMM16<br>IND16, X<br>IND16, Y<br>IND16, Z<br>EXT | 80<br>90<br>A0<br>2780<br>2790<br>27A0<br>37B0<br>37C0<br>37D0<br>37E0<br>37F0 | ff<br>ff<br>ff<br>—<br>—<br>—<br>jj kk<br>gggg<br>gggg<br>gggg<br>hh ll | 6<br>6<br>6<br>6<br>6<br>6<br>4<br>6<br>6<br>6<br>6 | — | — | — | — | Δ | Δ | Δ | Δ |
| SUBE | Subtract from E | (E) – (M : M + 1) ⇒ E | IMM16<br>IND16, X<br>IND16, Y<br>IND16, Z<br>EXT | 3730<br>3740<br>3750<br>3760<br>3770 | jj kk<br>gggg<br>gggg<br>gggg<br>hh ll | 4<br>6<br>6<br>6<br>6 | — | — | — | — | Δ | Δ | Δ | Δ |
| SWI | Software Interrupt | (PK : PC) + 2 ⇒ PK : PC<br>Push (PC)<br>(SK : SP) – 2 ⇒ SK : SP<br>Push (CCR)<br>(SK : SP) – 2 ⇒ SK : SP<br>$0 ⇒ PK<br>SWI Vector ⇒ PC | INH | 3720 | — | 16 | — | — | — | — | — | — | — | — |
| SXT | Sign Extend B into A | If B7 = 1<br>then A = $FF<br>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 | — | — | — | — | — | — | — | — |

## Instruction Set Summary (Continued)

| Mnemonic | Operation | Description | Address | Instruction | | | Condition Codes | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Mode | Opcode | Operand | Cycles | 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<br>(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]<br>Sign Extend AM | (D) ⇒ AM[15:0]<br>(E) ⇒ AM[31:16]<br>AM[35:32] = AM31 | INH | 27B1 | — | 4 | — | 0 | — | 0 | — | — | — | — |
| TEKB | Transfer EK to B | $0 ⇒ B[7:4]<br>(EK) ⇒ B[3:0] | INH | 27BB | — | 2 | — | — | — | — | — | — | — | — |
| TEM | Transfer E to AM[31:16]<br>Sign Extend AM<br>Clear AM LSB | (E) ⇒ AM[31:16]<br>$00 ⇒ AM[15:0]<br>AM[35:32] = AM31 | INH | 27B2 | — | 4 | — | 0 | — | 0 | — | — | — | — |
| TMER | Transfer AM to E Rounded | Rounded (AM) ⇒ Temp<br>If (SM • (EV + MV))<br>then Saturation ⇒ E<br>else Temp[31:16] ⇒ E | INH | 27B4 | — | 6 | — | Δ | — | Δ | Δ | Δ | — | — |
| TMET | Transfer AM to E Truncated | If (SM • (EV + MV))<br>then Saturation ⇒ E<br>else AM[31:16] ⇒ E | INH | 27B5 | — | 2 | — | — | — | — | Δ | Δ | — | — |
| TMXED | Transfer AM to IX : E : D | AM[35:32] ⇒ IX[3:0]<br>AM35 ⇒ IX[15:4]<br>AM[31:16] ⇒ E<br>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]<br>$0 ⇒ B[7:4] | INH | 37AF | — | 2 | | | | | — | — | — | — |
| TST | Test for Zero or Minus | (M) – $00 | IND8, X<br>IND8, Y<br>IND8, Z<br>IND16, X<br>IND16, Y<br>IND16, Z<br>EXT | 06<br>16<br>26<br>1706<br>1716<br>1726<br>1736 | ff<br>ff<br>ff<br>gggg<br>gggg<br>gggg<br>hh ll | 6<br>6<br>6<br>6<br>6<br>6<br>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<br>IND16, Y<br>IND16, Z<br>EXT | 2706<br>2716<br>2726<br>2736 | gggg<br>gggg<br>gggg<br>hh ll | 6<br>6<br>6<br>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]<br>(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]<br>(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 | Instruction | | | Condition Codes | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Mode | 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]<br>(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. The 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.
4. Cycle times for conditional branches are shown in "taken, not taken" order.

## Instruction Set Abbreviations and Symbols

| | | | | | |
|---|---|---|---|---|---|
| A | — | Accumulator A | X | — | Register used in operation |
| AM | — | Accumulator M | M | — | Address of one memory byte |
| B | — | Accumulator B | M +1 | — | Address of byte at M + $0001 |
| CCR | — | Condition code register | M : M + 1 | — | Address of one memory word |
| D | — | Accumulator D | (...)$_X$ | — | Contents of address pointed to by IX |
| E | — | Accumulator E | (...)$_Y$ | — | Contents of address pointed to by IY |
| EK | — | Extended addressing extension field | (...)$_Z$ | — | Contents of address pointed to by IZ |
| IR | — | MAC multiplicand register | E, X | — | IX with E offset |
| HR | — | MAC multiplier register | E, Y | — | IY with E offset |
| IX | — | Index register X | E, Z | — | IZ with E offset |
| IY | — | Index register Y | EXT | — | Extended |
| IZ | — | Index register Z | EXT20 | — | 20-bit extended |
| K | — | Address extension register | IMM8 | — | 8-bit immediate |
| PC | — | Program counter | IMM16 | — | 16-bit immediate |
| PK | — | Program counter extension field | IND8, X | — | IX with unsigned 8-bit offset |
| SK | — | Stack pointer extension field | IND8, Y | — | IY with unsigned 8-bit offset |
| SL | — | Multiply and accumulate sign latch | IND8, Z | — | IZ with unsigned 8-bit offset |
| SP | — | Stack pointer | IND16, X | — | IX with signed 16-bit offset |
| XK | — | Index register X extension field | IND16, Y | — | IY with signed 16-bit offset |
| YK | — | Index register Y extension field | IND16, Z | — | IZ with signed 16-bit offset |
| ZK | — | Index register Z extension field | IND20, X | — | IX with signed 20-bit offset |
| XMSK | — | Modulo addressing index register X mask | IND20, Y | — | IY with signed 20-bit offset |
| YMSK | — | Modulo addressing index register Y mask | IND20, Z | — | IZ with signed 20-bit offset |
| S | — | Stop disable control bit | INH | — | Inherent |
| MV | — | AM overflow indicator | IXP | — | Post-modified indexed |
| H | — | Half carry indicator | REL8 | — | 8-bit relative |
| EV | — | AM extended overflow indicator | REL16 | — | 16-bit relative |
| N | — | Negative indicator | b | — | 4-bit address extension |
| Z | — | Zero indicator | ff | — | 8-bit unsigned offset |
| V | — | Two's complement overflow indicator | gggg | — | 16-bit signed offset |
| C | — | Carry/borrow indicator | hh | — | High byte of 16-bit extended address |
| IP | — | Interrupt priority field | ii | — | 8-bit immediate data |
| SM | — | Saturation mode control bit | jj | — | High byte of 16-bit immediate data |
| PK | — | Program counter extension field | kk | — | Low byte of 16-bit immediate data |
| — | — | Bit not affected | ll | — | Low byte of 16-bit extended address |
| Δ | — | Bit changes as specified | mm | — | 8-bit mask |
| 0 | — | Bit cleared | mmmm | — | 16-bit mask |
| 1 | — | Bit set | rr | — | 8-bit unsigned relative offset |
| M | — | Memory location used in operation | rrrr | — | 16-bit signed relative offset |
| R | — | Result of operation | xo | — | MAC index register X offset |
| S | — | Source data | yo | — | MAC index register Y offset |
| | | | z | — | 4-bit zero extension |
| + | — | Addition | • | — | AND |
| - | — | Subtraction or negation (two's complement) | + | — | Inclusive OR (OR) |
| * | — | Multiplication | ⊕ | — | Exclusive OR (EOR) |
| / | — | Division | $\overline{\text{NOT}}$ | — | Complementation |
| > | — | Greater | : | — | Concatenation |
| < | — | Less | ⇒ | — | Transferred |
| = | — | Equal | ⇔ | — | Exchanged |
| ≥ | — | Equal or greater | ± | — | Sign bit; also used to show tolerance |
| ≤ | — | Equal or less | « | — | Sign extension |
| ≠ | — | Not equal | % | — | Binary value |
| | | | $ | — | Hexadecimal value |

## 2.7 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.

### 2.7.1 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 instruction 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. There are 52 predefined or reserved vectors, and 200 user-defined vectors.

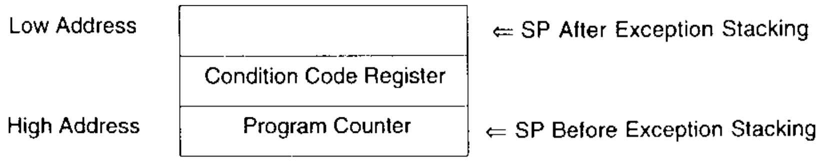
Each vector is assigned an 8-bit number. Vector numbers for some exceptions are generated by external devices; others are supplied by the processor. There is a direct mapping of vector number to vector table address. The CPU16 left shifts the vector number one place (multiplies by two) to convert it to an address.

**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 | Breakpoint |
| 5 | 000A | D | Bus Error |
| 6 | 000C | D | Software Interrupt Instruction (SWI) |
| 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 |

### 2.7.2 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. The following figure shows the exception stack frame.



### 2.7.3 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 reset 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.

### 2.7.4 Types of Exceptions

Exceptions can be generated either internally or externally. External exceptions are defined as asynchronous, and include interrupts, bus errors, breakpoints, and resets. Internal exceptions are defined as synchronous, and include the software interrupt (SWI) instruction, the background (BGND) instruction, illegal instruction exceptions, and the divide-by-zero exception. Refer to **3 Single-Chip Integration Module** for more information about resets and interrupts.

Asynchronous exceptions occur without reference to CPU16 or IMB clocks, but exception processing is synchronized. For all asynchronous exceptions but resets, exception processing begins at the first instruction boundary following recognition of an exception.

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 stacked return PK : PC value for 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. 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. $0002 is added to the PK : PC value before it is stacked.

### 2.7.5 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. Note that priority governs the order in which exception processing occurs, not the order in which exception handlers are executed.

Unless bus error, breakpoint, or reset occur during exception processing, the first instruction of all exception handler routines is guaranteed to execute before another exception is processed. Because interrupt exceptions have higher priority than synchronous exceptions, the first instruction in an interrupt handler is executed before other interrupts are sensed.

Bus error, breakpoint, and reset exceptions that occur during exception processing of a previous exception are processed before the first instruction of that exception's handler routine. The converse is not true. If an interrupt occurs during bus error exception processing, for example, the first instruction of the bus error handler is executed before interrupts are sensed. This permits the exception handler to mask interrupts during execution.

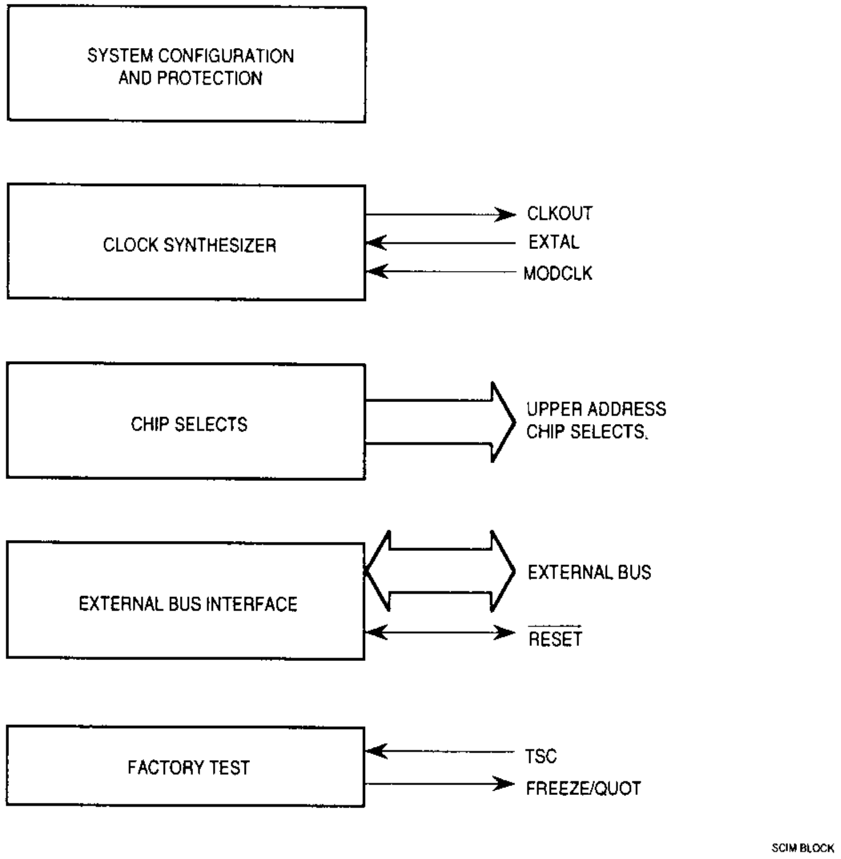
### 2.7.6 RTI Instruction

The return-from-interrupt instruction (RTI) must be the last instruction in all exception handlers except for 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 a reset initializes the stack pointer and does not create a stack frame.

# 3 Single-Chip Integration Module

The single-chip integration module (SCIM) consists of six submodules that, with a minimum of external devices, control system startup, initialization, configuration, and external bus. Refer to the following block diagram of the SCIM.



**Single-Chip Integration Module Block Diagram**

## 3.1 Overview

The system configuration and protection block controls MCU configuration and operating mode. The block also provides bus and software watchdog monitors.

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

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

The chip-select block provides eight general-purpose chip-select signals, a boot ROM chip-select signal, and two emulation-support chip-select signals. The general-purpose and boot ROM chip-select signals have associated base address registers and option registers.

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.

### SCIM Address Map

| Address | 15 8 | 7 0 |
|---|---|---|
| $YFFA00 | SCIM MODULE CONFIGURATION (SCIMCR) | |
| $YFFA02 | FACTORY TEST (SCIMTR) | |
| $YFFA04 | CLOCK SYNTHESIZER CONTROL (SYNCR) | |
| $YFFA06 | UNUSED | RESET STATUS REGISTER (RSR) |
| $YFFA08 | MODULE TEST E (SCIMTRE) | |
| $YFFA0A | PORT A DATA REGISTER (PORTA) | PORT B DATA REGISTER (PORTB) |
| $YFFA0C | PORT G DATA REGISTER (PORTG) | PORT H DATA REGISTER (PORTH) |
| $YFFA0E | PORT G DATA DIRECTION (DDRG) | PORT H DATA DIRECTION (DDRH) |
| $YFFA10 | UNUSED | PORT E DATA (PORTE0) |
| $YFFA12 | UNUSED | PORT E DATA (PORTE1) |
| $YFFA14 | PORT A/B DATA DIRECTION (DDRAB) | PORT E DATA DIRECTION (DDRE) |
| $YFFA16 | UNUSED | PORT E PIN ASSIGNMENT (PEPAR) |
| $YFFA18 | UNUSED | PORT F DATA (PORTF0) |
| $YFFA1A | UNUSED | PORT F DATA (PORTF1) |
| $YFFA1C | UNUSED | PORT F DATA DIRECTION (DDRF) |
| $YFFA1E | UNUSED | PORT F PIN ASSIGNMENT (PFPAR) |
| $YFFA20 | UNUSED | SYSTEM PROTECTION CONTROL (SYPCR) |
| $YFFA22 | PERIODIC INTERRUPT CONTROL (PICR) | |
| $YFFA24 | PERIODIC INTERRUPT TIMING (PITR) | |
| $YFFA26 | UNUSED | SOFTWARE SERVICE (SWSR) |
| $YFFA28 | UNUSED | PORT F EDGE-DETECT ENABLE (PORTFE) |
| $YFFA2A | UNUSED | PORT F INTERRUPT VECTOR (PFIVR) |
| $YFFA2C | UNUSED | PORT F INTERRUPT LEVEL (PFLVR) |
| $YFFA2E | UNUSED | UNUSED |
| $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 REGISTER (DREG) | |
| $YFFA3C | UNUSED | UNUSED |
| $YFFA3E | UNUSED | UNUSED |
| $YFFA40 | UNUSED | PORT C DATA (PORTC) |
| $YFFA42 | UNUSED | UNUSED |

## SCIM 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 | UNUSED | |
| $YFFA52 | UNUSED | |
| $YFFA54 | UNUSED | |
| $YFFA56 | UNUSED | |
| $YFFA58 | CHIP-SELECT BASE 3 (CSBAR3) | |
| $YFFA5A | CHIP-SELECT OPTION 3 (CSOR3) | |
| $YFFA5C | UNUSED | |
| $YFFA5E | UNUSED | |
| $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 | UNUSED | UNUSED |
| $YFFA7A | UNUSED | UNUSED |
| $YFFA7C | UNUSED | UNUSED |
| $YFFA7E | UNUSED | UNUSED |

Y = M111, where M is the state of the modmap bit in SCIMCR. In an M68HC16 MCU, M must always be set to one.

### 3.2 System Configuration

The MCU can operate as a stand-alone device (single-chip modes), with a 24-bit external address bus and an 8-bit external data bus (partially expanded mode), or with a 24-bit external address bus and a 16-bit external data bus. However, because ADDR[23:20] are driven to the same logic state as ADDR19, the external bus is effectively only 20 bits wide. SCIM pins can be configured for use as I/O ports or programmable chip select signals (Refer to **3.8 Chip Selects** and **3.9 General-Purpose Input/Output** for more information). System configuration is determined by setting bits in the SCIM configuration register (SCIMCR), and by asserting MCU pins during reset.

**SCIMCR** — Single-Chip Integration Module Configuration Register **$YFFA00**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EXOFF | FRZSW | FRZBM | CPUD | SLVE | 0 | SHEN | | SUPV | MM | ABD | RWD | IARB | | | |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 1 | 1 | * | * | 0 | 0 | 0 | 1 | 1 | * | * | 1 | 1 | 1 | 1 |

* Reset state is mode-dependent. Refer to the following bit descriptions.

The module configuration register controls system configuration. It can be read or written at any time, except for the module mapping (MM) bit, which must remain set to one.

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 continues to run.
1 = When FREEZE is asserted, the software watchdog is disabled.

FRZBM — Freeze Bus Monitor Enable

0 = When FREEZE is asserted, the periodic interrupt timer counters continue to run.
1 = When FREEZE is asserted, the periodic interrupt timer counters are disabled, preventing interrupts during software debug.

CPUD — CPU Development Support Disable

0 = Instruction pipeline signals available on pins IPIPE0 and IPIPE1
1 = Pins IPIPE0 and IPIPE1 placed in high-impedance state unless a breakpoint occurs

CPUD is reset to zero when the MCU is in an expanded mode, and to one in single-chip mode.

SLVE — Slave Mode Enable

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

This bit is a read-only status bit that reflects the state of DATA11 during reset. Slave mode is used for factory testing. Reset state is the complement of DATA11 during reset in fully expanded mode.

SHEN[1:0] — Show Cycle Enable

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

| 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 |

SUPV — Supervisor/Unrestricted Data Space

In systems that support restricted access, the SUPV bit places SCIM global registers in either supervisor data space or user data space. Because the CPU16 operates only in supervisory mode, SUPV has no effect.

MM — Module Mapping

0 = Internal modules are addressed from \$7FF000 – \$7FFFFF.

1 = Internal modules are addressed from \$FFF000 – \$FFFFFF.

The logic state of MM determines the value of ADDR23 in the IMB module address. Because ADDR[23:20] are driven to the same state as ADDR19, MM must be set to one. If MM is cleared, IMB modules are inaccessible. This bit can be written only once after reset.

ABD — Address Bus Disable

0 = Pins ADDR[2:0] operate normally.

1 = Pins ADDR[2:0] are disabled.

ABD is reset to zero when the MCU is in an expanded mode, and to one in single-chip mode. ABD can be written only once after reset.

RWD — Read/Write Disable

0 = R/$\overline{\text{W}}$ signal operates normally

1 = R/$\overline{\text{W}}$ signal placed in high-impedance state.

RWD is reset to zero when the MCU is in an expanded mode, and to one in single-chip mode. RWD can be written only once after reset.

IARB[3:0] — Interrupt Arbitration

Each module that can generate interrupts, including the SCIM, has an IARB field. Each IARB field can be assigned a value from \$0 to \$F. During an interrupt acknowledge cycle, IARB permits arbitration among simultaneous interrupts of the same priority level. The reset value of the SCIM IARB field is \$F. This prevents SCIM interrupts from being discarded. Initialization software must set the IARB field to a lower value if lower priority interrupts are to be arbitrated.

### 3.2.1 Operating Modes

During reset, the SCIM configures itself according to the states of the DATA, $\overline{\text{BERR}}$, MODCLK, and $\overline{\text{BKPT}}$ pins. DATA[11:0] provide pin configuration information. $\overline{\text{BERR}}$, MODCLK, and $\overline{\text{BKPT}}$ determine basic operation.

The SCIM can be configured to operate in one of three modes: 16-bit expanded, 8-bit expanded, and single chip. Operating mode is determined by the value of the DATA1 and $\overline{\text{BERR}}$ signals coming out of reset.

**Basic Configuration Options**

| Select Pin | Default Function<br>(Pin Left High) | Alternate Function<br>(Pin Pulled Low) |
|---|---|---|
| MODCLK | Synthesized System Clock | External System Clock |
| $\overline{\text{BKPT}}$ | Background Mode Disabled | Background Mode Enabled |
| $\overline{\text{BERR}}$ | Expanded Mode | Single-Chip Mode |
| DATA1 (if $\overline{\text{BERR}}$ = 1) | 8-Bit Expanded Mode | 16-Bit Expanded Mode |

$\overline{\text{BERR}}$, $\overline{\text{BKPT}}$, and MODCLK do not have internal pull-ups and must be driven to the desired state during reset.

Operating mode determines which address and data bus lines are used and which general-purpose I/O ports are available. The following table is a summary of bus and port configuration.

**Bus and Port Configuration Options**

| Mode | Address Bus | Data Bus | I/O Ports |
|---|---|---|---|
| 16-Bit Expanded | ADDR[18:3] | DATA[15:0] | — |
| 8-Bit Expanded | ADDR[18:3] | DATA[15:8] | DATA[7:0] = Port H |
| Single Chip | None | None | ADDR[18:11] = Port A<br>ADDR[10:3] = Port B<br>DATA[15:8] = Port G<br>DATA[7:0] = Port H |

Many pins on the MC68HC16X1, including data and address bus pins, have multiple functions. Reset value for these pins depends on operating mode. In expanded mode, the values of DATA[11:0] during reset determine the function of these pins. The functions of some pins can be changed by writing to the appropriate pin assignment register. Data bus pins have internal pull-ups and must be pulled low to achieve the alternate configuration desired. The following tables contain a summary of pin configuration options for each operating mode.

### 3.2.2 16-Bit Expanded Mode

In 16-bit expanded mode, (BERR = 1, DATA1 = 0) pins ADDR[18:3] and DATA[15:0] are configured as address and data pins, respectively. The alternate functions for these pins as ports A, B, G, and H are unavailable.

**16-Bit Expanded Mode Reset Configuration**

| Pin(s) Affected | Select Pin | Default Function (Pin Held High) | Alternate Function (Pin Held Low) |
|---|---|---|---|
| BR/CS0<br>FC0/CS3/PC0<br>FC1/PC1<br>FC2/CS5/PC2 | DATA2 | CS0<br>CS3<br>FC1<br>CS5 | BR<br>FC0<br>FC1<br>FC2 |
| ADDR19/CS6/PC3<br>ADDR23/CS10/ECLK | DATA3[1]<br>DATA7[1] | CS6<br>CS10 | ADDR19<br>ADDR23 |
| DSACK1/PE1<br>DS/PE4<br>AS/PE5<br>SIZ0/PE6<br>SIZ1/PE7 | DATA8 | DSACK1<br>DS<br>AS<br>SIZ0<br>SIZ1 | PE1<br>PE4<br>PE5<br>PE6<br>PE7 |
| MODCLK/PF0<br>IRQ[7:6]/PF[7:6] | DATA9 | MODCLK<br>IRQ[7:6] | PF0<br>PF[7:6] |
| BGACK/CSE<br>BG/CSM | DATA10 | BGACK<br>BG | CSE[2]<br>CSM[3] |
| Slave Mode | DATA11 | Slave Mode Disabled[4] | Slave Mode Enabled[4] |
| Emulation Mode (SCIM) | DATA10 | Disabled | Enabled |
| Emulation Mode (MRM) | DATA10, DATA13[5] | Disabled | Enabled |
| STOP Mode (MRM) | DATA14 | Array Enabled[6] | Array Disabled |
| STOP Mode (BEFLASH) | DATA15 | Array Enabled[7] | Array Disabled |

Notes

1. CS[7:9] outputs are not available on the MC68HC16X1. Corresponding pin assignment register fields (CSPA1[3:1]) are affected by the state of DATA[6:4] during reset.
2. CSE is enabled when DATA10 and DATA1 = 0 during reset.
3. CSM is enabled when DATA13, DATA10 and DATA1 = 0 during reset.
4. Slave mode used for factory test only.
5. Both mode select pins must be low to enable emulation (ROM) mode.
6. Driven to put ROM in STOP Mode. STOP mode disabled when DATA14 is held high.
7. Driven to put BEFLASH in STOP mode. STOP mode disabled when DATA15 is held high and STOP shadow bit is cleared.

### 3.2.3 8-Bit Expanded Mode

In 8-bit expanded mode (BERR = 1, DATA1 = 1), pins DATA[7:0] are configured as an 8-bit I/O port. Pins DATA[15:8] are configured as data pins. Pins ADDR[18:3] are configured as address pins. Emulator mode is always disabled.

**8-Bit Expanded Mode Reset Configuration**

| Pin(s) Affected | Select Pin | Default Function (Pin Held High) | Alternate Function (Pin Held Low) |
|---|---|---|---|
| BR/CS0<br>FC0/CS3/PC0<br>FC1/PC1<br>FC2/CS5/PC2 | N/A[1] | CS0<br>CS3<br>FC1<br>CS5 | CS0<br>CS3<br>FC1<br>CS5 |
| ADDR19/CS6/PC0<br>ADDR23/CS10/ECLK | N/A[1] | CS6<br>CS10 | CS6<br>CS10 |
| DSACK1/PE1<br>DS/PE4<br>AS/PE5<br>SIZ0/PE6<br>SIZ1/PE7 | DATA8 | DSACK1<br>DS<br>AS<br>SIZ0<br>SIZ1 | PE1<br>PE4<br>PE5<br>PE6<br>PE7 |
| MODCLK/PF0<br>IRQ[7:6]/PF[7:6] | DATA9 | MODCLK<br>IRQ[7:6] | PF0<br>PF[7:6] |
| BGACK/CSE<br>BG/CSM | N/A[1] | BGACK<br>BG | BGACK<br>BG |

NOTES:

[1] These pins have only one reset configuration in 8-bit expanded mode.

### 3.2.4 Single-Chip Mode

In single-chip mode, when BERR = 0 during reset, ADDR[18:3] are configured as 8-bit I/O ports (port A and port B) and DATA[15:0] are configured as 8-bit I/O ports (port G and port H). There is no external data bus path. Expanded mode configuration options are not available: I/O ports A, B, E, F, G, and H are always selected. BERR can be tied low permanently to select single-chip mode.

### Single-Chip Mode Reset Configuration

| Pin(s) Affected | Function |
|---|---|
| ADDR[18:11] | PA[7:0] |
| ADDR[10:3] | PB[7:0] |
| BR/CS0 | CS0 |
| FC0/CS3/PC0<br>FC1/PC1<br>FC2/CS5/PC2<br>ADDR19/CS6/PC3 | PC[3:0] |
| ADDR23/CS10/ECLK | — |
| DSACK1/PE1<br>DS/PE4<br>AS/PE5<br>SIZ0/PE6<br>SIZ1/PE7 | PE[7:4], PE1 |
| MODCLK/PF0<br>IRQ[7:6]/PF[7:6] | PF0<br>PF[7:6] |
| DATA[15:8] | PG[7:0] |
| DATA[7:0] | PH[7:0] |
| BGACK/CSE<br>BG/CSM | BGACK<br>BG |

### 3.2.5 Emulation Support

The SCIM contains logic that can be used to replace on-chip ports externally. The SCIM also contains special support logic that allows external emulation of internal ROM. This emulation support feature enables the development of a single-chip application in expanded mode.

Emulator mode is a special type of 16-bit expanded operation. It is entered by holding DATA10 low, BERR high, and DATA1 low during reset. In emulator mode, all port A, B, E, G, and H data and data direction registers and the port E pin assignment register are mapped externally. Port C data, port F data and data direction registers, and port F pin assignment register are accessible normally in emulator mode.

An emulator chip select (CSE) is asserted whenever any of the externally-mapped registers are addressed. The signal is asserted on the falling edge of AS. The SCIM does not respond to these accesses, allowing external logic, such as a port replacement unit (PRU) to respond. Accesses to externally mapped registers require three clock cycles.

External ROM emulation is enabled by holding DATA1, DATA10, and DATA13 low during reset (BERR must be held high during reset to enable the ROM module). While ROM emulation mode is enabled, memory chip select signal CSM is asserted whenever a valid access to an address assigned to the masked ROM array is made. The ROM module does not acknowledge IMB accesses while in emulation mode. This causes the SCIM to run an external bus cycle for each access. Refer to **3.8 Chip Selects** and **8 Masked ROM Module** for more information.

## 3.3 System Protection

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

**SYPCR** — System Protection Control Register **$YFFA21**

| 15 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| NOT USED | SWE | SWP | SWT | | HME | BME | BMT | |
| RESET: | | | | | | | | |
| | 1 | $\overline{\text{MODCLK}}$ | 0 | 0 | 0 | 0 | 0 | 0 |

The system protection control register controls system monitor functions, software watchdog clock prescaling, and bus monitor timing. In operating modes, this register can be written only once following power-on or reset, but can be read at any time. In test mode, it can be written at any time.

SWE — Software Watchdog Enable

0 = Software watchdog disabled
1 = Software watchdog enabled

SWP — Software Watchdog Prescale

This bit controls the value of the software watchdog prescaler.

0 = Software watchdog clock not prescaled
1 = Software watchdog clock prescaled by 512

The reset value of SWP is the complement of the state of the MODCLK pin during reset.

SWT[1:0] — Software Watchdog Timing

This field selects the divide ratio used to establish software watchdog timeout period. The following table gives the ratio for each combination of SWP and SWT bits.

| 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 a bus monitor timeout period as shown in the following table.

| BMT | Bus Monitor Timeout Period |
|---|---|
| 00 | 64 System Clocks |
| 01 | 32 System Clocks |
| 10 | 16 System Clocks |
| 11 | 8 System Clocks |

### 3.3.1 Bus Monitor

The internal bus monitor checks for excessively long response times during normal bus cycles (DSACK1) and during IACK cycles. The monitor asserts BERR if response time is excessive.

DSACK1 response times are measured in clock cycles. The maximum allowable response time can be selected by setting the BMT field.

The monitor does not check DSACK1 response on the external bus unless it initiates the bus cycle. The BME bit in the SYPCR enables the internal bus monitor for internal to external bus cycles. If a system contains external bus masters, an external bus monitor must be implemented, and the internal to external bus monitor option must be disabled.

### 3.3.2 Halt Monitor

The halt monitor responds to an assertion of HALT on the internal bus, caused by a double bus fault. This signal is asserted by the CPU after a double bus fault occurs. A flag in the reset status register (RSR) indicates that the last reset was caused by the halt monitor. The halt monitor reset can be inhibited by the DBE bit in the SYPCR.

### 3.3.3 Spurious Interrupt Monitor

The spurious interrupt monitor causes a bus error exception if no interrupt arbitration occurs during interrupt acknowledge cycle.

### 3.3.4 Software Watchdog

**SWSR** — Software Service Register **$YFFA27**

| 15 8 | 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|
| NOT USED | SWSR | | | | | | | |
| RESET: | | | | | | | | |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Register shown with read value

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

Perform a software watchdog service sequence as follows:

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

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

Watchdog clock rate is affected by SWP and SWT in SYPCR.

When SWT[1:0] are modified, a watchdog service sequence must be performed before the new timeout period will take effect.

The reset value of SWP is the complement of the state of the MODCLK pin on the rising edge of reset.

Software watchdog timeout period is given in the following equation:

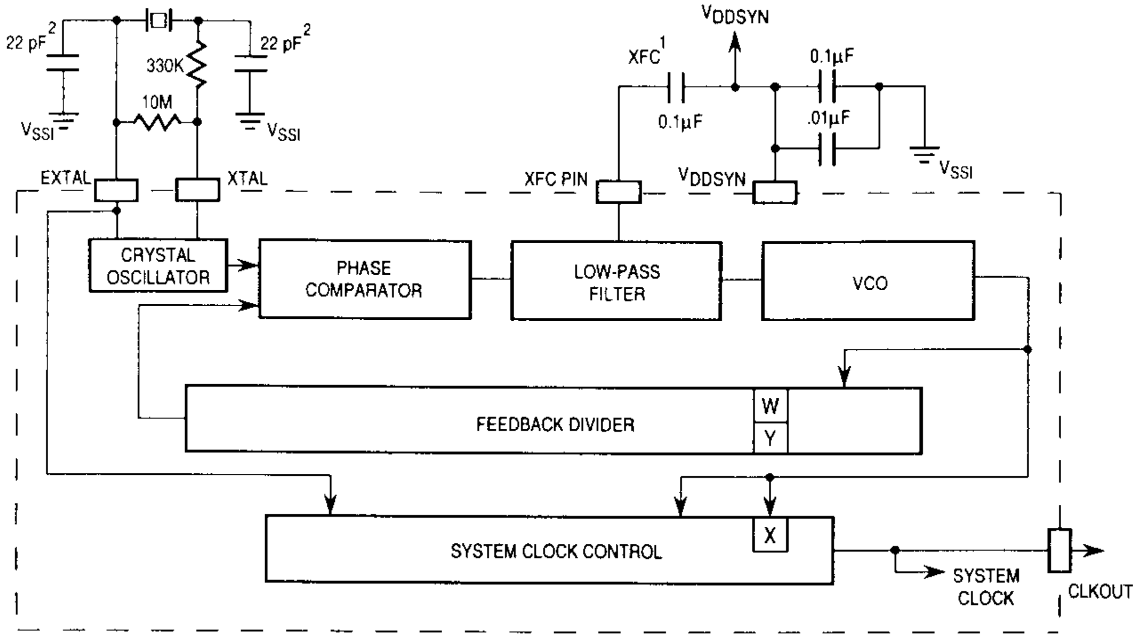
$$\text{TIMEOUT PERIOD} = \frac{\text{DIVIDE COUNT}}{\text{EXTAL FREQUENCY}}$$

### 3.4 System Clock

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

The system clock signal can be generated in three ways. Either an internal reference or an external reference, or an external clock signal can be input. Keep the distinction between an external reference and an external clock signal in mind while reading the rest of this section.

Following is a block diagram of the clock submodule.



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

SYS CLOCK BLOCK 32KHZ

**System Clock Block Diagram**

### 3.4.1 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 mode of operation. When MODCLK is held low during reset, the clock synthesizer is disabled and an external system clock signal must be applied — SYNCR control bits have no effect.

A reference crystal must be connected between the EXTAL and XTAL pins to use the internal oscillator. If either an external reference signal or an external system clock signal is applied through the EXTAL pin, the XTAL pin must be left floating. External reference signal frequency must be less than or equal to maximum specified reference frequency. External system clock signal frequency must be less than or equal to maximum specified system clock frequency.

When an external system clock signal is applied, the duty cycle of the input is critical, especially at operating frequencies close to maximum. The relationship between clock signal duty cycle and clock signal period is expressed as follows:

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

### 3.4.2 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 equal 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.

To maintain a 50% clock duty cycle, VCO frequency is either two or four times clock frequency, depending on the state of the X bit in SYNCR.

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

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, since PLL stability depends on the VCO, which uses this supply. 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 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. When X = 0 (reset state), the divider is enabled, and system clock frequency is one-fourth VCO frequency; setting X disables the divider, doubling clock speed without changing VCO speed. There is no VCO relock delay. The SYNCR W bit controls a three-bit prescaler in the feedback divider. Setting W increases VCO speed by a factor of four. The SYNCR Y field determines the count modulus for a modulo 64 down counter, causing it to divide by a value of Y + 1. When either the W or Y value change, there is a VCO relock delay.

Clock frequency is determined by SYNCR bit settings as follows:

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

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. The reset state of SYNCR ($3F00) produces a modulus-64 count. Maximum specified clock frequency with a 32.768-kHz reference is 16.78 kHz.

### 3.4.3 Loss of Clock

The SCIM can detect loss of either an external clock signal or a clock signal generated by the PLL. Two bits in SYNCR determine how the SCIM responds to the loss of a clock signal.

The loss-of-clock oscillator disable (LOSCD) bit enables (LOSCD = 0) or disables (LOSCD = 1) an internal RC oscillator which is used as the time base for the loss-of-clock detector, and also provides an alternate system clock signal. LOSCD must be cleared for loss-of-clock detection to take place.

The reset enable (RSTEN) bit determines how the SCIM responds when it detects the loss of a clock signal. LOSCD must be cleared for the RSTEN bit to have any effect. When the RSTEN bit is set and loss of clock is detected, the SCIM generates an asynchronous reset. If RSTEN is cleared when loss of clock is detected, the internal oscillator is used as system clock until edges are detected on the EXTAL input. All clock switching is done synchronously, so that no short pulses or glitches occur on the system clock.

LOSCD and RSTEN are automatically cleared during reset, ensuring that an alternate clock signal is available during reset. If the system clock fails during reset, the SCIM detects the condition, switches to the alternate clock, and completes reset processing.

An MCU using the alternate clock as the system clock is said to be operating in limp mode. The limp mode status bit (SLIMP) in SYNCR indicates whether the MCU is running in limp mode.

Loss of clock is recognized during low-power operation as well as normal operation, provided LOSCD is cleared. Low-power operation for loss of clock is the same as normal operation.

### 3.4.4 Clock Control

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

**SYNCR** — Clock Synthesizer Control Register **$YFFA04**

| 15 | 14 | 13 | | | | | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | X | Y | | | | | | EDIV | 0 | LOCSD | SLIMP | SLOCK | RSTEN | STSCIM | STEXT |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | U | U | 0 | 0 | 0 |

When the on-chip clock synthesizer is used, system clock frequency is controlled by the bits in the upper byte of SYNCR. Bits in the lower byte show the status of, or control the operation of, internal and external clocks. Because the CPU16 always operates in supervisor mode, SYNCR can be read or written at any time.

W — Frequency Control (VCO)
This bit controls a prescaler tap in the synthesizer feedback loop. Setting the bit increases the VCO speed by a factor of four. VCO relock delay is required.

X — Frequency Control Bit (Prescale)
This bit controls a divide by two prescaler that is not in the synthesizer feedback loop. Setting it doubles clock speed without changing VCO speed. There is no VCO relock delay.

Y[5:0] — Frequency Control (Counter)

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

EDIV — ECLK Divide Rate
0 = ECLK frequency is system clock divided by 8.
1 = ECLK frequency is system clock divided by 16.
ECLK is an external M6800 bus clock available on pin ADDR23. Refer to **3.8 Chip Selects** for more information.

LOCSD — Loss-of-Clock Oscillator Disable
0 = Enable the loss-of-clock oscillator.
1 = Disable the loss-of-clock oscillator.

SLIMP — Limp Mode Flag
0 = External crystal is VCO reference.
1 = Loss of crystal reference.
When the on-chip synthesizer is used, loss of reference frequency causes SLIMP to be set. The VCO continues to run using the base control voltage. Maximum limp frequency is the maximum specified system clock frequency. X-bit state affects limp frequency.

SLOCK — Synthesizer Lock Flag
0 = VCO is enabled, but has not locked.
1 = VCO has locked on the desired frequency (or system clock is external).
The MCU maintains reset state until the synthesizer locks, but SLOCK does not indicate synthesizer lock status until after the user writes to SYNCR.

RSTEN — Reset Enable
0 = Loss of crystal causes the MCU to operate in limp mode.
1 = Loss of crystal causes system reset.

STSCIM — Stop Mode SCIM Clock
0 = When LPSTOP is executed, the SCIM clock is driven from the crystal oscillator and the VCO is turned off to conserve power.
1 = When LPSTOP is executed, the SCIM clock is driven from the VCO.

STEXT — Stop Mode External Clock
0 = When LPSTOP is executed, the CLKOUT signal is held negated to conserve power.
1 = When LPSTOP is executed, the CLKOUT signal is driven from the SCIM clock, as determined by the state of the STSCIM bit.

### 3.4.5 Periodic Interrupt Timer

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

**PICR** — Periodic Interrupt Control Register **$YFFA22**

| | 15 | 14 | 13 | 12 | 11 | 10 | | 8 | 7 | | | | | | | 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 |

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

PIRQL[2:0] — Periodic Interrupt Request Level

The following table shows what interrupt request level is asserted when a periodic interrupt is generated. If a PIT interrupt and an external IRQ of the same priority occur simultaneously, the PIT interrupt is serviced first. The periodic timer continues to run when the interrupt is disabled.

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

PIV[7:0] — Periodic Interrupt Vector

The bits of this field contain the vector generated in response to an interrupt from the periodic timer. When the SCIM responds, the periodic interrupt vector is placed on the bus.

**PITR** — Periodic Interrupt Timer Register **$YFFA24**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | | | | | | | 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 |

PITR contains the count value for the periodic timer. A zero value turns off 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

The reset state of PTP is the complement of the state of the MODCLK signal during reset.

PITM[7:0] — Periodic Interrupt Timing Modulus Field

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

$$\text{PIT PERIOD} = \frac{[\text{PITM(PRESCALE)(4)}]}{\text{EXTAL}}$$

where

PIT Period = Periodic interrupt timer period

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

EXTAL = Crystal frequency

Prescale = 512 or 1 depending on the state of the PTP bit in the PITR

## 3.5 External Bus Interface

The external bus interface (EBI) transfers information between the internal MCU bus and external devices when the MCU is operating in expanded modes. In fully expanded mode, the external bus has 24 address lines and 16 data lines. In partially expanded mode, the external bus has 24 address lines and 8 data lines. Because the CPU16 drives only 20 of the 24 IMB address lines, ADDR[23:20] follow the output state of ADDR19.

The EBI supports byte, word, and long-word transfers. Ports are accessed through the use of asynchronous cycles controlled by the data transfer (SIZ1 and SIZ0) and the data size acknowledge pin, DSACK1.

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

To add flexibility and minimize the necessity for external logic, MCU chip select logic can be synchronized with EBI transfers. Chip select logic can also provide internally-generated bus control signals for these accesses. Refer to **3.8 Chip Selects** for more information.

### 3.5.1 Bus Control Signals

The CPU initiates a bus cycle by driving the address, size, function code, and read/write outputs. At the beginning of the cycle, size signals SIZ0 and SIZ1 are driven along with the function code signals. The size signals indicate the number of bytes remaining to be transferred during an operand cycle. They are valid while the address strobe (AS) is asserted. The following table shows SIZ0 and SIZ1 encoding. The read/write (R/W) signal determines the direction of the transfer during a bus cycle. This signal changes state, when required, at the beginning of a bus cycle, and is valid while AS is asserted. R/W only transitions when a write cycle is preceded by a read cycle or vice versa. The signal can remain low for two consecutive write cycles.

**Size Signal Encoding**

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

### 3.5.2 Function Codes

Function code signals FC[2:0] are automatically generated by the CPU16. The function codes can be considered address extensions that automatically select one of eight address spaces to which an address applies. These spaces are designated as either user or supervisor, and program or data spaces. Because the CPU16 always operates in supervisor mode (FC2 always = 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 AS is asserted.

**CPU16 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 |

### 3.5.3 Address Bus

Address bus signals ADDR[19:0] define the address of the most significant byte to be transferred during a bus cycle. The MCU places the address on the bus at the beginning of a bus cycle. The address is valid while $\overline{\text{AS}}$ is asserted. The CPU16 drives ADDR[23:20] to the same logic state as ADDR19.

### 3.5.4 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.

### 3.5.5 Data Bus

Data 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.

### 3.5.6 Data Strobe

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

### 3.5.7 Bus Cycle Termination Signals

During bus cycles, external devices assert a data transfer and size acknowledge signal, $\overline{\text{DSACK1}}$. During a read cycle, the signal tells the MCU to terminate the bus cycle and to latch data. During a write cycle, the signal indicates that an external device has successfully stored data and that the cycle can terminate. The $\overline{\text{DSACK1}}$ signal also indicates to the MCU the size of the port for the bus cycle just completed. In the MC68HC16X1, the $\overline{\text{DSACK0}}$ pin is not provided and an external device indicates the availability of data by asserting $\overline{\text{DSACK1}}$ regardless of port size.

The bus error ($\overline{\text{BERR}}$) signal is also a bus cycle termination indicator and can be used in the absence of $\overline{\text{DSACK1}}$ to indicate a bus error condition. $\overline{\text{BERR}}$ can also be asserted in conjunction with $\overline{\text{DSACK1}}$, provided $\overline{\text{BERR}}$ meets the appropriate timing requirements. The internal bus monitor can be used to generate the $\overline{\text{BERR}}$ signal for internal and internal-to-external transfers. When $\overline{\text{BERR}}$ is asserted, the CPU16 takes a bus error exception.

### 3.5.8 Data Transfer Mechanism

MCU architecture supports byte, word, and long-word operands, allowing access to 8- and 16-bit data ports through the use of asynchronous cycles. Because the $\overline{\text{DSACK0}}$ pin is not present on the MC68HC16X1 MCU, word transfers to or from an 8-bit port require the use of the STED and LDED instructions. Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for further information.

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 the following figure. OP0 is the most significant byte of a long-word operand, and OP3 is the least significant byte. The two bytes of a word-length operand are OP0 (most significant) and OP1. The single byte of a byte-length operand is OP0.

| **Operand** | **Byte Order** | | | |
|---|---|---|---|---|
| | 31 24 | 23 16 | 15 8 | 7 0 |
| Long Word | OP0 | OP1 | OP2 | OP3 |
| Three Byte | | OP0 | OP1 | OP2 |
| Word | | | OP0 | OP1 |
| Byte | | | | OP0 |

**Operand Byte Order**

### 3.5.9 Operand Alignment

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

ADDR0 also affects the operation of the data multiplexer. During an operand transfer, ADDR[23:1] indicate the word base address of the portion of the operand to be accessed, and ADDR0 indicates the byte offset from the base. Bear in mind the fact that ADDR[23:20] are driven to the same logic state as ADDR19.

### 3.5.10 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.

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 software compatible with the M68HC11 CPU. The CPU16 treats misaligned long-word transfers as two misaligned word transfers.

### 3.5.11 Operand Transfer Cases

The following table shows 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.

**Operand Transfer Cases**

| Transfer Case | SIZ1 | SIZ0 | ADDR0 | DATA [15:8] | DATA [7:0] |
|---|---|---|---|---|---|
| Byte to 16-bit Port (Even) | 0 | 1 | 0 | OP0 | (OP0) |
| Byte to 16-bit Port (Odd) | 0 | 1 | 1 | (OP0) | OP0 |
| Word to 16-bit Port (Aligned) | 1 | 0 | 0 | OP0 | OP1 |
| Word to 16-bit Port (Misaligned) | 1 | 0 | 1 | (OP0) | OP0 |
| 3 Byte to 16-bit Port (Aligned)[2] | 1 | 1 | 0 | OP0 | OP1 |
| 3 Byte to 16-bit Port (Misaligned)[2] | 1 | 1 | 1 | (OP0) | OP0 |
| Long Word to 16-bit Port (Aligned) | 0 | 0 | 0 | OP0 | OP1 |
| Long Word to 16-bit Port (Misaligned)[3] | 1 | 0 | 1 | (OP0) | OP0 |

NOTES:
1. Operands in parentheses are ignored by the CPU16 during read cycles.
2. Three-byte transfer cases occur only as a result of a long word to byte transfer.
3. The CPU16 treats misaligned long-word transfers as two misaligned word transfers.

In the MC68HC16X1, in which the DSACK0 pin is not present, word transfers to or from an 8-bit port can be performed by using the STED and LDED instructions. In addition, byte reads of an 8-bit port can only access even addresses unless a chip select is used.

## 3.6 Resets

Reset procedures handle system initialization and recovery from catastrophic failure. The MCU performs resets with a combination of hardware and software. The SCIM 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.

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

Reset is the highest-priority CPU16 exception. Any processing in progress is aborted by the reset exception, and cannot be restarted. Only essential tasks are performed during reset exception processing. Other initialization tasks must be accomplished by the exception handler routine.

### 3.6.1 SCIM Reset Mode Selection

The logic states of certain MCU pins during reset determine SCIM operating configuration. Refer to **3.2.1 Operating Modes** for more information.

### 3.6.2 MCU Module Pin Function During Reset

Module pins usually 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. For more information, refer to the sections of this technical summary that present information about the individual modules. The following table is a summary of module pin functions out of reset.

**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 | MOSI | Discrete Input |
| | MISO | Discrete Input |
| | SCK | Discrete Input |
| | PQS3 | Discrete Input |
| | PQS2 | Discrete Input |
| | PQS1 | Discrete Input |
| | PQS0/$\overline{\text{SS}}$ | Discrete Input |
| | TXD | Discrete Input |
| | RXD | RXD |

### 3.6.3 Reset Timing

The $\overline{\text{RESET}}$ input must be asserted for a specified minimum period in order for reset to occur. 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) to protect write cycles from being aborted by reset. While $\overline{\text{RESET}}$ is asserted, SCIM 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 reset input pin goes to an inactive, high-impedance state for 10 cycles. At the end of this 10-cycle period, the reset input is tested. When the input is at logic level one, reset exception processing begins. If, however, the reset input is at logic level zero, the reset control logic drives the pin low for another 512 cycles. At the end of this period, the pin again goes to high-impedance state for 10 cycles, then it is tested again. The process repeats until $\overline{\text{RESET}}$ is released.

### 3.6.4 Power-On Reset

When the SCIM 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 $V_{DDSYN}$, so that the MCU can 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.

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

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

### 3.6.5 Use of Three-State Control Pin

Asserting the three-state control (TSC) input causes all MCU output drivers to go to an inactive, high-impedance condition. Although TSC is an active-high input, it does not have an internal pull-down and must be tied low when not in use.

TSC must remain asserted for ten system clock cycles 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 ten clock 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 ten 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.

## 3.7 Interrupts

Interrupt recognition and servicing involve complex interaction between the central processing unit, the single-chip integration module, and a device or module requesting interrupt service.

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. The CPU16 handles interrupts as a type of asynchronous exception.

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.

Note that on the MC68HC16X1 the only external interrupts available are $\overline{\text{IRQ6}}$ and $\overline{\text{IRQ7}}$.

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 a wired-NOR. Simultaneous requests with different priorities 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 through the external bus interface and SCIM interrupt control logic. The CPU treats external interrupt requests as though they had come from the SCIM.

External $\overline{\text{IRQ6}}$ is an active-low level-sensitive input. External $\overline{\text{IRQ7}}$ is an active-low transition-sensitive input. It requires both an edge and a voltage level for validity.

$\overline{\text{IRQ6}}$ is maskable. $\overline{\text{IRQ7}}$ is nonmaskable. The $\overline{\text{IRQ7}}$ input is transition-sensitive to prevent redundant servicing and stack overflow. A nonmaskable interrupt is generated each time $\overline{\text{IRQ7}}$ is asserted, and each time the priority mask changes from %111 to a lower number while $\overline{\text{IRQ7}}$ is asserted.

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

The 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.

### 3.7.1 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.

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 latched into the CCR IP field to mask lower-priority interrupts during exception processing, and it is decoded by modules that have requested interrupt service to determine whether the current interrupt acknowledge cycle pertains to them.

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

Each module that can make an interrupt service request, including the SCIM, 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 SCIM IARB value is used for arbitration between internal and external interrupt requests. The reset value of IARB for the SCIM is %1111. The reset IARB value for all other modules is %0000. Initialization software must assign different IARB values to implement an arbitration scheme.

Each module must have a unique IARB value. 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 requests service. This point is important for two reasons: the CPU interrupt acknowledge cycle is not driven on the external bus unless the SCIM wins contention, and failure to contend causes an interrupt acknowledge bus cycle to be terminated by a bus error, which causes a spurious interrupt exception to be taken.

When arbitration is complete, the dominant module must place an interrupt vector number on the data bus and terminate the bus cycle. 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 bus cycle termination signals. If the device does not respond in time, a spurious interrupt exception is taken.

The periodic interrupt timer (PIT) in the SCIM 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 **3.4.5 Periodic Interrupt Timer** for more information.

### 3.7.2 Interrupt Processing

The following summary outlines the interrupt processing sequence. When the sequence begins, a valid interrupt service request has been detected and is pending.

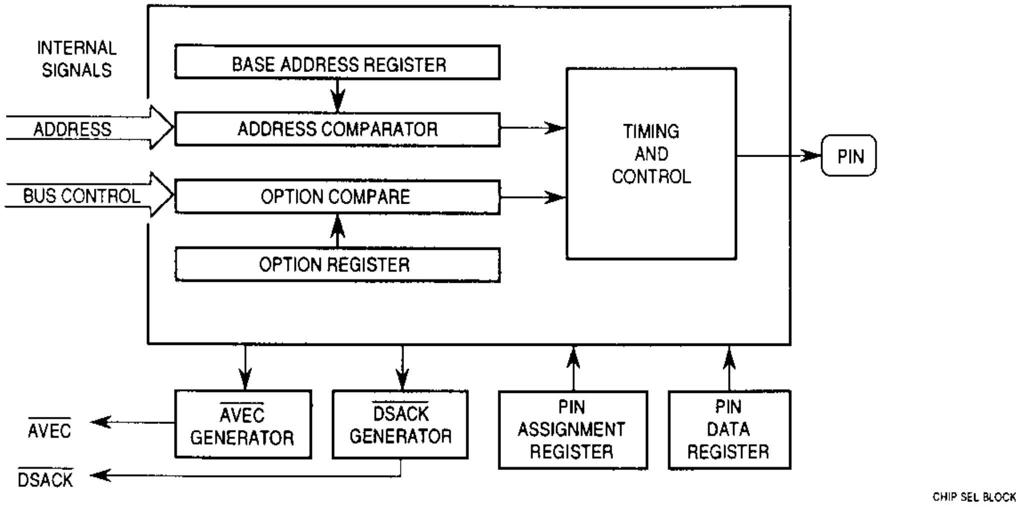
A. The CPU finishes higher priority exception processing or reaches an instruction boundary.
B. Processor state is stacked, then the CCR PK extension field is cleared.
C. The interrupt acknowledge cycle begins:
   1. FC[2:0] are driven to %111 (CPU space) encoding.
   2. The address bus is driven as follows. ADDR[23:20] = %1111; ADDR[19:16] = %1111, which indicates that the cycle is an interrupt acknowledge CPU space cycle; ADDR[15:4] = %111111111111; 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 $\overline{\text{DSACK1}}$ signals appropriate to the access. The CPU16 acquires the vector number.
   2. Chip-select logic asserts $\overline{\text{AVEC}}$ internally 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.
E. The vector number is converted to a vector address.
F. The content of the vector address is loaded into the PC, and the processor transfers control to the exception handler routine.

### 3.8 Chip Selects

Typical microcontrollers require additional hardware to provide external chip-select signals. The MC68HC16X1 includes five general-purpose programmable chip select circuits that can provide 2 to 13 clock cycle access to external memory and peripherals. Two additional chip select signals, CSE and CSM, provide emulation support. Address block sizes of 2 Kbytes to 1 Mbyte can be selected. However, because ADDR[23:20] are driven to the same logic state as ADDR19, 512-Kbyte blocks are the largest usable size.

Chip select assertion can be synchronized with bus control signals to provide output enable, read/write strobes, or interrupt acknowledge signals. Logic can also generate DSACK and AVEC signals internally. A single DSACK generator is shared by all circuits. Multiple chip selects assigned to the same address and control must have the same number of wait states. Chip selects 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, a chip select signal is asserted. Select signals are active low. The following block diagram shows a single chip-select circuit.



**Chip-Select Circuit Block Diagram**

If a chip select function is given the same address as a microcontroller module or memory array, an access to that address goes to the module or array, and the chip select signal is not asserted.

Each chip-select pin has two or more functions. Configuration out of reset is determined by operating mode. In single-chip mode, all chip select pins except CS10 and CS0 are configured for alternate functions or discrete output. In expanded modes, appropriate pins are configured for chip select operation, but chip select signals cannot be asserted until a transfer size is chosen. In fully expanded mode, data bus pins can be held low to enable alternate chip-select pin functions .

The following table shows allocation of chip selects and discrete outputs to MCU pins.

**Chip Select Pin Allocation**

| Chip Select Function | Alternate Function | Discrete Outputs Function |
|---|---|---|
| $\overline{\text{CS0}}$ | $\overline{\text{BR}}$ | — |
| $\overline{\text{CSM}}$ | $\overline{\text{BG}}$ | — |
| $\overline{\text{CSE}}$ | $\overline{\text{BGACK}}$ | — |
| $\overline{\text{CS3}}$ | FC0 | PC0 |
| — | FC1 | PC1 |
| $\overline{\text{CS5}}$ | FC2 | PC2 |
| $\overline{\text{CS6}}$ | ADDR19 | PC3 |
| $\overline{\text{CS10}}$ | ADDR23 | ECLK |

### 3.8.1 Emulation Mode Chip Select Signals

Emulation mode chip select signals are used during external register or ROM emulation. Pin function is controlled by a chip select pin assignment register, but the other chip select registers do not affect these signals.

During emulation mode operation, all port A, B, E, G, and H data and data direction registers, and the port E pin assignment register are mapped externally. The emulation chip select signal ($\overline{\text{CSE}}$) is asserted when any of these registers is addressed. The SCIM does not respond to these accesses. An external device, such as a port replacement unit, can respond instead. Refer to **3.2.5 Emulation Support** for more information.

An internal module chip select signal ($\overline{\text{CSM}}$) can also be enabled during emulator mode operation. When the ROM module is enabled, $\overline{\text{CSM}}$ is asserted when an access to an address assigned to the masked ROM array is made and the STOP bit is clear. This allows an external device to emulate the ROM. Internal $\overline{\text{DSACK}}$ is generated by the ROM module after it has inserted the number of wait states specified by the WAIT field in the MRMCR. Refer to **8 Masked ROM Module** for more information.

### 3.8.2 Chip Select Registers

Pin assignment registers (CSPAR) determine functions of chip select pins. Pin assignment registers also determine port size (8- or 16-bit) for dynamic bus allocation.

A pin data register (PORTC) latches discrete output data.

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). However, because the logic state of ADDR20 is always the same as the state of ADDR19, the largest usable block size is 512 Kbytes. Address blocks for separate chip-select functions can overlap.

Chip-select option registers (CSOR) 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 code often 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.

### 3.8.3 Pin Assignment Registers

The pin assignment registers contain pairs of bits that determine the functions of chip-select pins. Pin functions are shown in the tables following the register diagrams.

Reset state of the pin assignment registers depends on operating mode. In the register diagrams, reset values are shown in the following order: single-chip mode, partially expanded mode, and fully expanded mode. The notation DATA# indicates that a bit goes to the logic level of that data bus pin on reset. DATA lines have weak pull-ups. During reset in fully expanded mode, an active external device can pull the data lines low to select alternate functions.

**CSPAR0** — Chip Select Pin Assignment Register 0 **$YFFA44**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | CSPA0[6] | | 0 | FC1 | CSPA0[4] | | CSPA0[3] | | CSPA0[2] | | CSPA0[1] | | 0 | |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | DATA2 | 1 | 0 | 1 | DATA2 | 1 | $\overline{\text{DATA10}}$ | 1 | $\overline{\text{DATA10}}$ | 1 | DATA2 | 1 | 0 | 0 |

**CSPAR1** — Chip Select Pin Assignment Register 1 **$YFFA46**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | CSPA1[4] | | CSPA1[3] | | CSPA1[2] | | CSPA1[1] | | CSPA1[0] | |
| RESET: | | | | | | | | | | | | | | | |
| | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | 0 | 0 | 0 | 0 | 0 | DATA7 | 1 | DATA6 | 1 | DATA5 | 1 | DATA4 | 1 | DATA3 | 1 |

Clearing both CS10 select bits (CSPAR1[9:8]) enables the M6800 bus clock (ECLK) on ADDR23.

**Pin Assignment Field Encoding**

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

**CSPAR0 Pin Functions**

| CSPAR0 Field | Chip-Select Signal | Alternate Signal | Discrete Output |
|---|---|---|---|
| CSPA0[6] | $\overline{\text{CS5}}$ | FC2 | PC2 |
| CSPA0[5] | — | FC1 | PC1 |
| CSPA0[4] | $\overline{\text{CS3}}$ | FC0 | PC0 |
| CSPA0[3] | $\overline{\text{CSE}}$ | $\overline{\text{BGACK}}$ | — |
| CSPA0[2] | $\overline{\text{CSM}}$ | $\overline{\text{BG}}$ | — |
| CSPA0[1] | $\overline{\text{CS0}}$ | $\overline{\text{BR}}$ | — |

CSPAR1 Pin Functions

| CSPAR1 Field | Chip-Select Signal | Alternate Signal | Discrete Output |
|---|---|---|---|
| CSPA1[4] | CS10 | ADDR23 | ECLK |
| CSPA1[3] | — | — | — |
| CSPA1[2] | — | — | — |
| CSPA1[1] | — | — | — |
| CSPA1[0] | CS6 | ADDR19 | PC3 |

A pin programmed as a discrete output drives an external signal to the value specified in the port C data register (PORTC), with the following exceptions:

a. No discrete output function is available on pins BR, BG, or BGACK.
b. ADDR23 provides ECLK output rather than a discrete output signal.

Internal chip select logic is inhibited when discrete output or alternate function are assigned.

Port size is determined when a pin is assigned as a chip select. When a pin is assigned to an 8-bit port, the chip select is asserted at all addresses within the block range. If a pin is assigned to a 16-bit port, the upper/lower byte field of the option register selects the byte with which the chip select is associated.

### 3.8.4 Base Address Registers

A base address is the starting address for the block enabled by a given chip select. Block size determines the extent of the block above the base address. Each chip select has an associated base register so that an efficient address map can be constructed for each application. If a chip select is assigned an address used by a microcontroller module, the module has priority. The chip select does not respond to an access.

**CSBARBT** — Chip Select Base Address Register Boot ROM **$YFFA48**

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

**CSBAR[0:10]** — Chip Select Base Address Registers **$YFFA4C–$YFFA74**

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

ADDR[23:20] is at the same logic level as ADDR19 during internal CPU master operation. ADDR[23:20] must match ADDR19 for the chip select to be active.

ADDR[23:11] — Base Address Field

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

Since ADDR20 = ADDR19 in the CPU16, the maximum block size is 512 Kbytes. Because ADDR[23:20] follow the logic state of ADDR19, if all 24 address lines are used, addresses from $080000 to $F7FFFF are inaccessible.

BLKSZ — Block Size Field

This field determines the size of the block above the base address that must be enabled by the chip select. The following table shows bit encoding for the base address registers block size field.

| Block Size Field | 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] |

During normal operation ADDR[23:20] is at the same logic level as ADDR19.

### 3.8.5 Option Registers

The option registers contain eight fields that determine timing of and conditions for assertion of chip select signals. These make the chip selects useful for generating peripheral control signals. Certain constraints set by fields in the base address register and in the option register must be satisfied to assert a chip select signal and to provide $\overline{\text{DSACK}}$ or autovector support.

**CSORBT** — Chip Select Option Register Boot ROM **$YFFA4A**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | | | 6 | 5 | 4 | 3 | | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MODE | BYTE | | R/$\overline{\text{W}}$ | | STRB | $\overline{\text{DSACK}}$ | | | | SPACE | | IPL | | | $\overline{\text{AVEC}}$ |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

**CSOR[0:10]** — Chip Select Option Registers **$YFFA4E–$YFFA76**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | | | 6 | 5 | 4 | 3 | | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MODE | BYTE | | R/$\overline{\text{W}}$ | | STRB | $\overline{\text{DSACK}}$ | | | | SPACE | | IPL | | | $\overline{\text{AVEC}}$ |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

MODE — Asynchronous/Synchronous Mode

0 = Asynchronous mode selected
1 = Synchronous mode selected

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

In synchronous mode, the $\overline{\text{DSACK}}$ field is not used because a bus cycle is only performed as a synchronous operation. When a match condition occurs on a chip select programmed for synchronous operation, the chip select signals the EBI that an E-clock cycle is pending.

BYTE — Upper/Lower Byte Option

This field is used only when the chip select 16-bit port option is selected in the pin assignment register. The following table lists upper/lower byte options.

| Byte | Description |
|---|---|
| 00 | Disable |
| 01 | Lower Byte |
| 10 | Upper Byte |
| 11 | Both Bytes |

R/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. The following table shows the options.

| R/W | Description |
|---|---|
| 00 | Reserved |
| 01 | Read Only |
| 10 | Write Only |
| 11 | Read/Write |

STRB — Address Strobe/Data Strobe

0 = Address strobe<br>
1 = Data strobe

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

DSACK — Data and Size Acknowledge

This field specifies the source of DSACK in asynchronous mode. It also allows the user to adjust bus timing with internal DSACK generation by controlling the number of wait states that are inserted to optimize bus speed in a particular application. The following table shows the DSACK field encoding. A no-wait encoding (%0000) corresponds to a three clock-cycle bus. The fast termination encoding (%1110) corresponds to a two clock-cycle bus. Microcontroller modules typically respond at this rate, but fast termination can also be used to access fast external memory.

| DSACK | Description |
|---|---|
| 0000 | No Wait States |
| 0001 | 1 Wait State |
| 0010 | 2 Wait States |
| 0011 | 3 Wait States |
| 0100 | 4 Wait States |
| 0101 | 5 Wait States |
| 0110 | 6 Wait States |
| 0111 | 7 Wait States |
| 1000 | 8 Wait States |
| 1001 | 9 Wait States |
| 1010 | 10 Wait States |
| 1011 | 11 Wait States |
| 1100 | 12 Wait States |
| 1101 | 13 Wait States |
| 1110 | Fast Termination |
| 1111 | External DSACK |

SPACE — Address Space Select

This option field is used to select an address space to be used by the chip select logic. The CPU16 normally operates in supervisor space. All space types can be used. Interrupt acknowledge cycles take place in CPU space.

| Space Field | Address Space |
|---|---|
| 00 | CPU Space |
| 01 | User Space |
| 10 | Supervisor Space |
| 11 | Supervisor/User Space |

IPL — Interrupt Priority Level

When the SPACE field is set for CPU space (%00), chip select logic can be used for interrupt acknowledge. During an interrupt acknowledge cycle, the priority level on address lines ADDR[3:1] is compared to the value in the IPL field. If the values are the same, then a chip select can be asserted, provided other option register conditions are met. When the SPACE field has any value except %00, the IPL field determines whether an access takes place in program or data space. The following table shows IPL field encoding.

| IPL | Space = 00 | Space = 01, 10, 11 |
|---|---|---|
| 000 | All | Data or Program |
| 001 | IPL1 | Data |
| 010 | IPL2 | Program |
| 011 | IPL3 | Reserved |
| 100 | IPL4 | Reserved |
| 101 | IPL5 | Data |
| 110 | IPL6 | Program |
| 111 | IPL7 | Reserved |

This field only affects the response of chip selects and does not affect interrupt recognition by the CPU. **All** means that a chip select signal is asserted regardless of the priority of the interrupt.

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

0 = External interrupt vector enabled
1 = Autovector enabled

This field selects one of two methods of acquiring the interrupt vector during the interrupt acknowledge cycle. It is not usually used with a chip select pin.

If the chip select is configured to trigger on an interrupt acknowledge cycle (SPACE = %00) and the AVEC field is set to one, the chip select automatically generates an $\overline{\text{AVEC}}$ in response to the interrupt acknowledge cycle. Otherwise, the vector must be supplied by the requesting device.

**PORTC** — Port C Data Register **$YFFA41**

| 15 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| NOT USED | | 0 | PC6 | PC5 | PC4 | PC3 | PC2 | PC1 | PC0 |
| RESET: | | U | U | U | U | U | U | U | U |

PORTC controls the state of chip-select pins programmed for discrete output. When a pin is assigned as a discrete output, the value in this register appears at the output. Bit 7 is not used. Writing to bit 7 has no effect, and it always reads zero.

### 3.8.6 Chip Select Reset Operation

The reset values of the chip select pin assignment fields in CSPAR0 and CSPAR1 depend on the operating mode selected. Refer to **3.2.1 Operating Modes** and to the discussion of these registers for more information.

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

## 3.9 General-Purpose Input/Output

The SCIM contains six general-purpose input/output ports: ports A, B, E, F, G, and H. (Port C, an output-only port, is included under the discussion of chip selects.) Ports A, B, and G are available in single-chip mode only and port H is available in single-chip or 8-bit expanded modes only. Ports E, F, G, and H have an associated data direction register (DDR) to configure each pin as input or output. Ports A and B share a DDR that configures each port as input or output. Ports E and F have associated pin assignment registers that configure each pin as digital I/O or an alternate function. Port F has an edge-detect flag register that indicates whether a transition has occurred on any of its pins.

The following table shows the shared functions of the general-purpose I/O ports and the modes in which they are available.

**General-Purpose I/O Ports**

| Port | Shared Function | Modes |
|---|---|---|
| A | ADDR[18:11] | Single Chip |
| B | ADDR[10:3] | Single Chip |
| E | Bus Control | All |
| F | IRQ[7:6]/MODCLK | All |
| G | DATA[15:8] | Single Chip |
| H | DATA[7:0] | Single Chip, 8-Bit Expanded |

Access to the port A, B, E, F, G, and H data and data direction registers, and the port C, E, and F pin assignment registers require three clock cycles to ensure timing compatibility with external port replacement logic. Port registers are byte-addressable and are grouped to allow coherent word access to port data register pairs A-B and G-H, as well as word-aligned long word coherency of A-B-G-H port data registers. Port registers are not affected by CPU reset.

If emulation mode is enabled, the emulation mode chip-select signal $\overline{\text{CSE}}$ is asserted whenever an access to ports A, B, E, G, and H data and data direction registers or the port E pin assignment register is made. The SCIM does not respond to these accesses, but allows external logic, such as a Motorola port replacement unit (PRU) MC68HC33 to respond. Port C data and data direction register, port F data and data direction register, and the port F pin assignment register remain accessible.

A write to the port A, B, E, F, G, or H data register is stored in the internal data latch. If any port pin is configured as an output, the value stored for that bit is driven on the pin. A read of the port 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.

### 3.9.1 Ports A and B

Ports A and B are available in single-chip mode only. One data direction register controls data direction for both ports. Port A and B registers can be read or written at any time the MCU is not in emulator mode.

**PORTA** — Port A Data Register **$YFFA0A**
**PORTB** — Port B Data Register **$YFFA0B**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PA7 | PA6 | PA5 | PA4 | PA3 | PA2 | PA1 | PA0 | PB7 | PB6 | PB5 | PB4 | PB3 | PB2 | PB1 | PB0 |
| RESET: | | | | | | | | | | | | | | | |
| U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

**DDRAB** — Port A/B Data Direction Register **$YFFA14**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | DDA | DDB | DDRE | | | | | | | |
| RESET: | | | | | | | | | | | | | | | |
| U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

DDA and DDB control the direction of the pin drivers for ports A and B, respectively, when the pins are configured for I/O. Setting DDA or DDB configures all pins in the corresponding port as outputs. Clearing DDA or DDB to zero configures all pins in the corresponding port as inputs.

### 3.9.2 Port E

Port E can be made available in all operating modes. The state of $\overline{\text{BERR}}$ and DATA8 during reset controls whether the port E pins are used as bus control signals or discrete I/O lines.

If the MCU is in emulator mode, an access of the port E data, data direction, or pin assignment registers (PORTE, DDRE, PEPAR) is forced to go external. This allows port replacement logic to be supplied externally, giving an emulator access to the bus control signals.

**PORTE** — Port E Data Register **$YFFA11, $YFFA13**

| 15 | | | | | | | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NOT USED | | | | | | | | PE7 | PE6 | PE5 | PE4 | PE3 | PE2 | PE1 | PE0 |
| RESET: | | | | | | | | | | | | | | | |
| | | | | | | | | U | U | U | U | U | U | U | U |

PORTE is a single register that can be accessed in two locations. It can be read or written at any time the MCU is not in emulator mode.

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

| 15 | | | | | | | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DDRAB | | | | | | | | DDE7 | DDE6 | DDE5 | DDE4 | DDE3 | DDE2 | DDE1 | DDE0 |
| RESET: | | | | | | | | | | | | | | | |
| U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

The bits in this register control the direction of the pin drivers when the pins are configured as I/O. In this register, setting a bit configures the corresponding pin as an output. Clearing a bit configures the corresponding pin as an input. DDRE can be read or written at any time the MCU is not in emulator mode.

The bits in this register control the direction of the pin drivers when the pins are configured as I/O. Any bit in this register set to one configures the corresponding pin as an output. Any bit in this register cleared to zero configures the corresponding pin as an input. This register can be read or written at any time the MCU is not in emulator mode.

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

| 15 | | | | | | | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NOT USED | | | | | | | | PEPA7 | PEPA6 | PEPA5 | PEPA4 | PEPA3 | PEPA2 | PEPA1 | PEPA0 |
| RESET: | | | | | | | | | | | | | | | |
| | | | | | | | | DATA8 | DATA8 | DATA8 | DATA8 | DATA8 | DATA8 | DATA8 | DATA8 |
| | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

The bits in this register control the function of each port E pin. Any bit set to one defines the corresponding pin to be a bus control signal, with the function shown in the following table. Any bit cleared to zero defines the corresponding pin to be an I/O pin, controlled by PORTE and DDRE. The reset state of PEPAR in expanded mode is shown first, the reset state in single-chip mode is shown second.

**Port E Pin Assignments**

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

$\overline{\text{BERR}}$ and DATA8 control the state of this register following reset. If $\overline{\text{BERR}}$ and/or DATA8 are low during reset, this register is set to $00, defining all port E pins as I/O pins. If $\overline{\text{BERR}}$ and DATA8 are both high during reset, the register is set to $FF, which defines all port E pins as bus control signals.

### 3.9.3 Port F

Port F pins can be configured as interrupt request inputs, edge-detect input/outputs, or discrete input/outputs. When port F pins are configured for edge detection, and a priority level is specified by writing a value to the port F edge-detect interrupt level register (PFLVR), port F control logic generates an interrupt request when the specified edge is detected. Interrupt vector assignment is made by writing a value to the port F edge-detect interrupt vector register (PFIVR). The edge-detect interrupt has the lowest arbitration priority in the SCIM.

**PORTF** — Port F Data Register **$YFFA19, $YFFA1B**

| 15 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| NOT USED | | PF7 | PF6 | PF5 | PF4 | PF3 | PF2 | PF1 | PF0 |
| RESET: | | | | | | | | | |
| | | U | U | U | U | U | U | U | U |

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

Port F is a single register that can be accessed in two locations. It can be read or written at any time, including when the MCU is in emulator mode.

**DDRF** — Port F Data Direction Register **$YFFA1D**

| 15 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| NOT USED | | DDF7 | DDF6 | DDF5 | DDF4 | DDF3 | DDF2 | DDF1 | DDF0 |
| RESET: | | | | | | | | | |
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

The bits in this register control the direction of port F pin drivers when the pins are configured for I/O. Setting any bit in this register configures the corresponding pin as an output. Clearing any bit in this register configures the corresponding pin as an input.

**PFPAR** — Port F Pin Assignment Register **$YFFA1F**

| 15 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| NOT USED | | PFPA3 | | NOT USED | | NOT USED | | PFPA0 | |
| RESET: | | | | | | | | | |
| | | DATA9 | DATA9 | | | | | DATA9 | DATA9 |
| | | 0 | 0 | | | | | 0 | 0 |

The fields in this register determine the functions of pairs of port F pins as shown in the following tables. $\overline{\text{BERR}}$ and DATA9 determine the reset state of this register. If $\overline{\text{BERR}}$ and/or $\overline{\text{DATA9}}$ are low during reset, this register is set to $00, defining all port F pins to be I/O pins. If $\overline{\text{BERR}}$ and DATA9 are both high during reset, the register is set to $FF, which defines all port F pins, except PF0, as interrupt signals. The reset state of PFPAR in expanded mode is shown first, the reset state in single-chip mode is shown second.

### Port F Pin Assignments

| PFPAR Field | Port F Signal | Alternate Signal |
|---|---|---|
| PFPA3** | PF[7:6] | $\overline{\text{IRQ}}$[7:6] |
| PFPA0** | PF[1:0] | MODCLK* |

*MODCLK signal is only recognized during reset.
**PF[5:2] are not connected. These bits have no meaning.

### PFPAR Pin Functions

| PFPAx Bits | Port F Signal |
|---|---|
| 00 | I/O pin without edge detect |
| 01 | Rising edge detect |
| 10 | Falling edge detect |
| 11 | Interrupt request |

**PORTFE** — Port F Edge-Detect Flag Register **$YFFA2B**

| 15 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| NOT USED | | EF7 | EF6 | EF5 | EF4 | EF3 | EF2 | EF1 | EF0 |
| RESET: | | | | | | | | | |
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

When the corresponding pin is configured for edge detection, a PORTFE bit is set if an edge is detected. PORTFE bits remain set, regardless of the subsequent state of the corresponding pin, until cleared. To clear a bit, first read PORTFE, then write the bit to zero. When a pin is configured for general-purpose I/O or for use as an interrupt request input, PORTFE bits do not change state.

**PFIVR** — Port F Edge-Detect Interrupt Vector Register **$YFFA2B**

| 15 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| NOT USED | | PFIVR7 | PFIVR6 | PFIVR5 | PFIVR4 | PFIVR3 | PFIVR2 | PFIVR1 | PFIVR0 |
| RESET: | | | | | | | | | |
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

This register determines which vector in the exception vector table is used for interrupts generated by the port F edge-detect logic. Program PFIVR[7:0] to the value pointing to the appropriate interrupt vector. Refer to **2 Central Processing Unit** for interrupt vector assignments.

**PFLVR** — Port F Edge-Detect Interrupt Level Register **$YFFA2D**

| 15 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| NOT USED | | 0 | 0 | 0 | 0 | 0 | PFLV2 | PFLV1 | PFLV0 |
| RESET: | | | | | | | | | |
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

PFLVR determines the priority level of the port F edge-detect interrupt. The reset value is $00, indicating that the interrupt is disabled. When several sources of interrupts from the SCIM are arbitrating for the same level, the port F edge-detect interrupt has the lowest arbitration priority.

### 3.9.4 Port G

Port G is available in single-chip mode only. These pins are always configured for use as general-purpose I/O in single-chip mode.

### 3.9.5 Port H

Port H is available in single-chip and 8-bit expanded modes only. The function of these pins is determined by the operating mode. There is no pin assignment register associated with this port.

**PORTG** — Port G Data Register **$YFFA0C**
**PORTH** — Port H Data Register **$YFFA0D**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PG7 | PG6 | PG5 | PG4 | PG3 | PG2 | PG1 | PG0 | PH7 | PH6 | PH5 | PH4 | PH3 | PH2 | PH1 | PH0 |
| RESET: | | | | | | | | | | | | | | | |
| U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

These port data registers can be read or written any time the MCU is not in emulation mode. Reset has no effect.

**DDRG** — Port G Data Direction Register **$YFFA0E**
**DDRH** — Port H Data Direction Register **$YFFA0F**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DDG7 | DDG6 | DDG5 | DDG4 | DDG3 | DDG2 | DDG1 | DDG0 | DDH7 | DDH6 | DDH5 | DDH4 | DDH3 | DDH2 | DDH1 | DDH0 |
| RESET: | | | | | | | | | | | | | | | |
| U | U | U | U | U | U | U | U | U | U | U | U | U | U | U | U |

The bits in this register control the direction of the port pin drivers when pins are configured as I/O. Setting a bit configures the corresponding pin as an output. Clearing a bit configures the corresponding pin as an input.

### 3.10 Factory Test

Test functions are integrated into the SCIM to support scan-based testing of the various MCU modules during production. Test submodule registers are intended for Motorola use. Register names and addresses are provided to show the user that these addresses are occupied.

**SCIMTR** — Single-Chip Integration Module Test Register **$YFFA02**

**SCIMTRE** — Single-Chip Integration Module Test Register (E Clock) **$YFFA08**

**TSTMSRA** — Master Shift Register A **$YFFA30**

**TSTMSRB** — Master Shift Register B **$YFFA32**

**TSTSC** — Test Module Shift Count **$YFFA34**

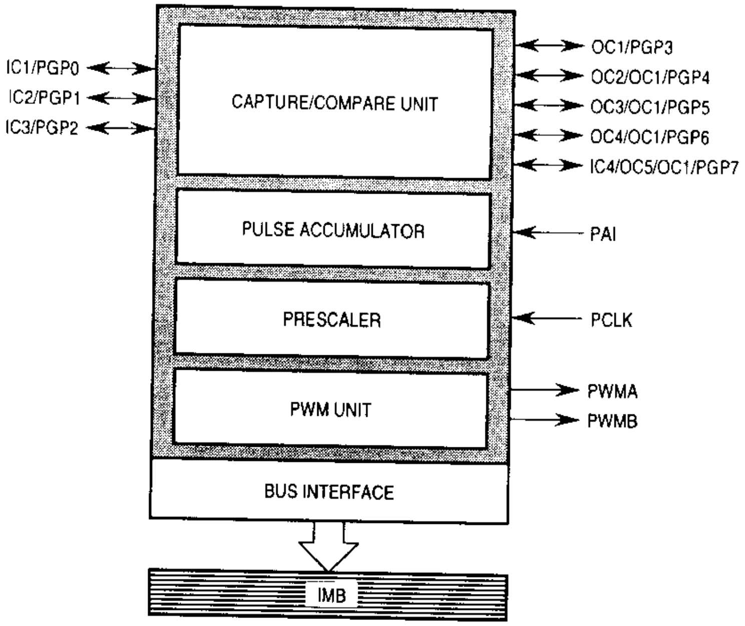
**TSTRC** — Test Module Repetition Count **$YFFA36**

**CREG** — Test Submodule Control Register **$YFFA38**

**DREG** — Distributed Register **$YFFA3A**

# 4 General-Purpose Timer Module

The 11-channel general-purpose timer (GPT) is used in systems where a moderate level of CPU control is required. 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).



GPT BLOCK

**GPT Block Diagram**

## 4.1 Overview

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.

GPT input capture/output compare pins are bidirectional and can be used to form an 8-bit parallel port. The pulse-width modulator outputs can be used as general-purpose outputs. The PAI and PCLK inputs can be used as general-purpose inputs.

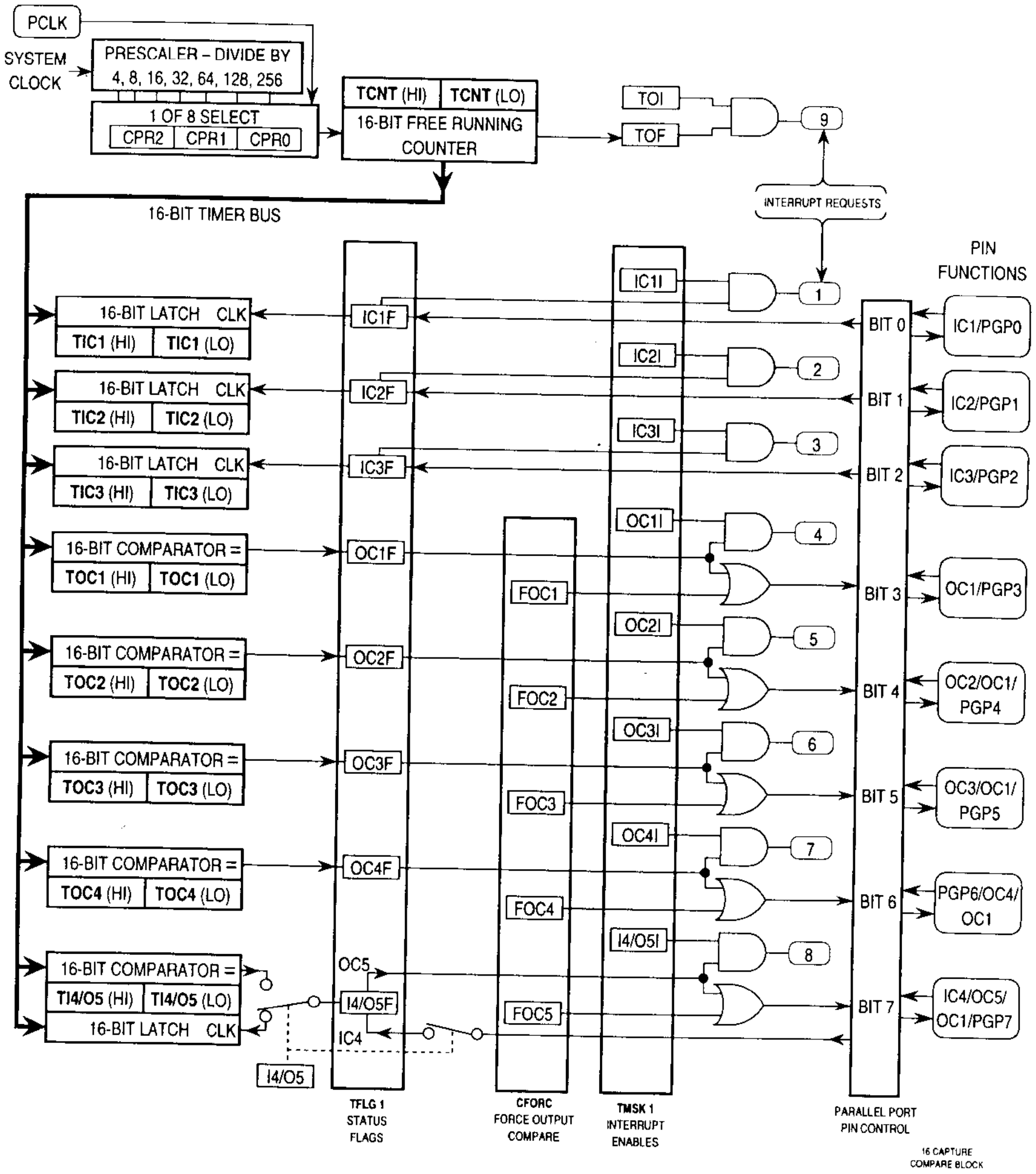
**GPT Address Map**

| Address | 15 8 | 7 0 |
|---|---|---|
| $YFF900 | GPT MODULE CONFIGURATION (GPTMCR) | |
| $YFF902 | (RESERVED FOR TEST) | |
| $YFF904 | INTERRUPT CONFIGURATION (ICR) | |
| $YFF906 | 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) | |
| $YFF92E–<br>$YFF93F | RESERVED | |

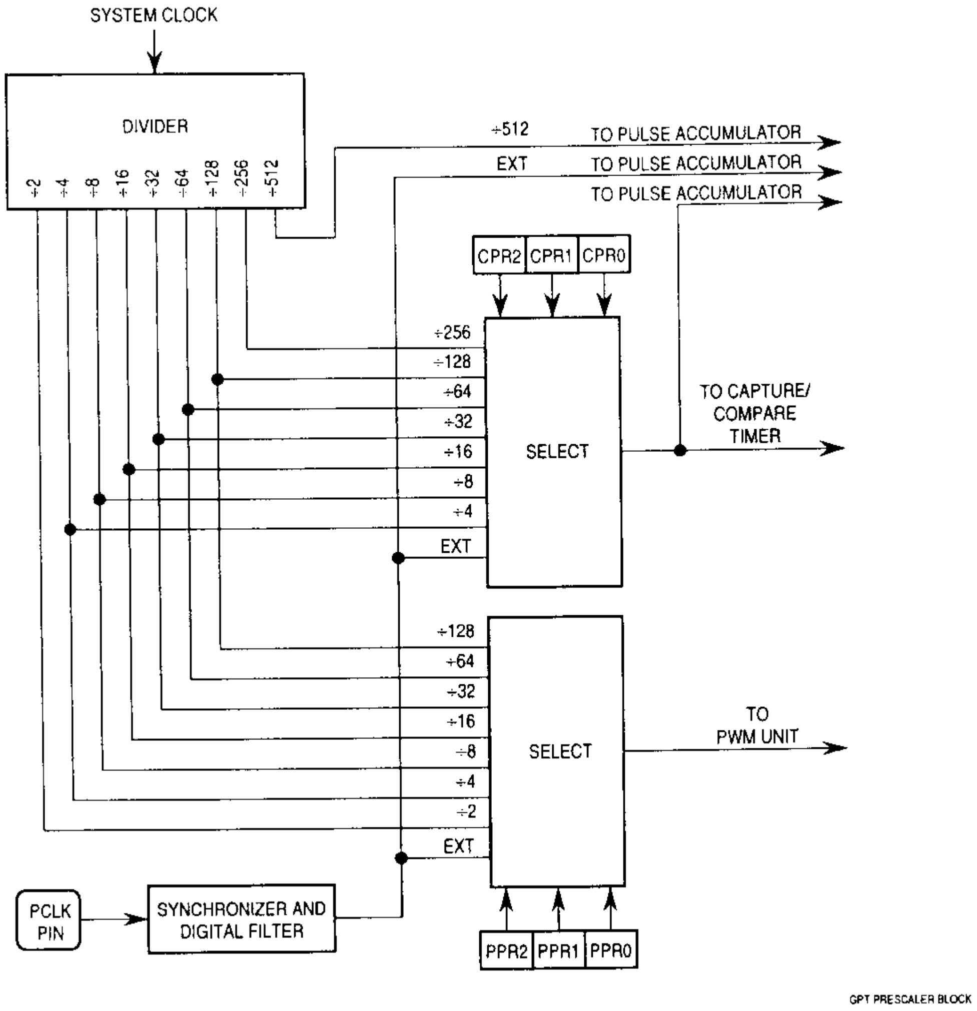
Y = M111, where M is the state of the modmap bit in SCIMCR. In an M68HC16 MCU, M must always be set to one.

## 4.2 Capture/Compare Unit

The capture/compare unit features three input capture channels, four output compare channels, and one input capture/output compare channel (function selected via control register). These channels share a 16-bit free-running counter (TCNT) which derives its clock from seven stages of a 9-stage prescaler or from external clock input PCLK. The pulse accumulator logic includes its own 8-bit counter and can operate in either event counting mode or gated time accumulation mode. Block diagrams of the GPT timer and prescaler follow.



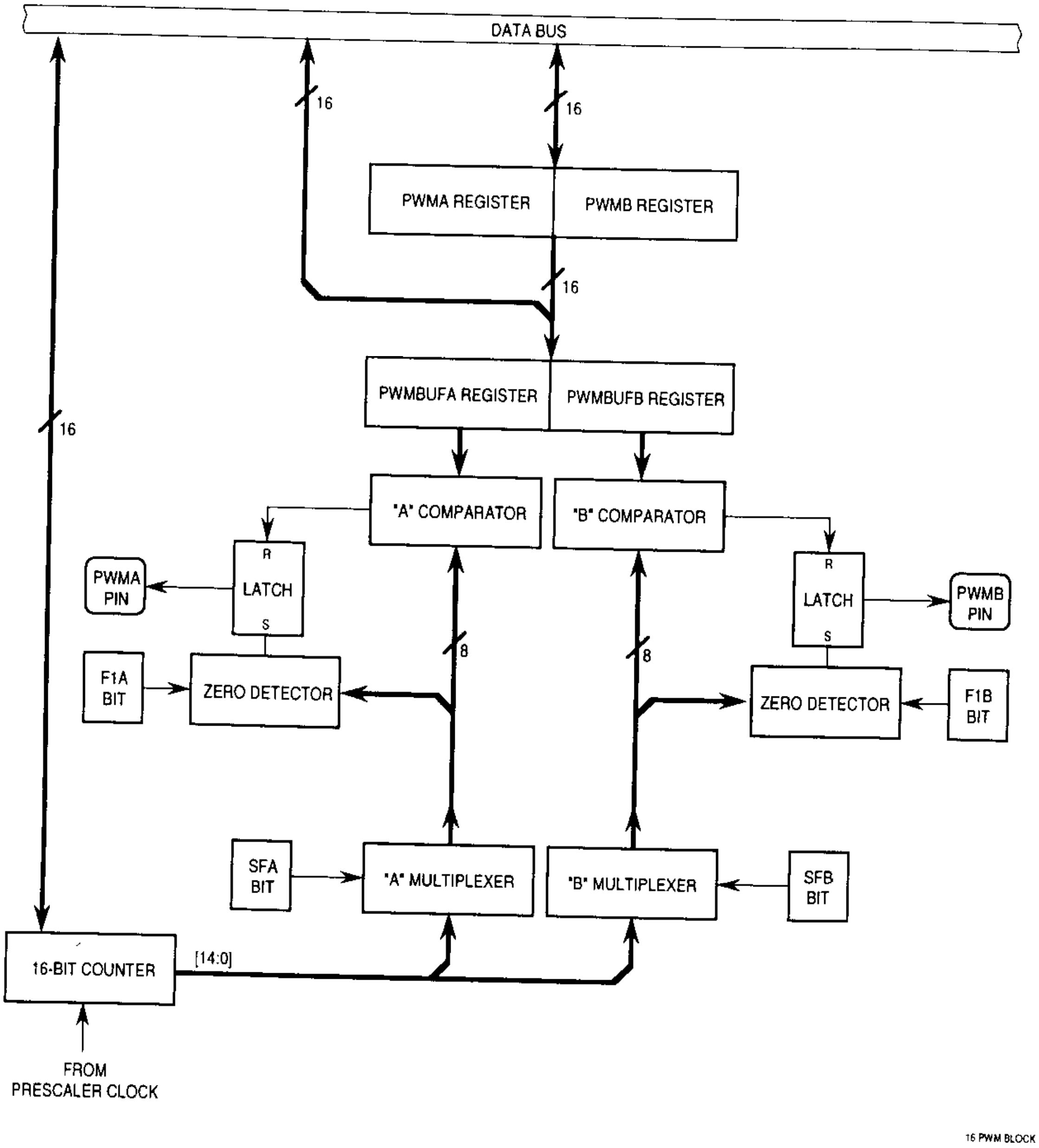
**Timer Diagram**



**Prescaler Block Diagram**

## 4.3 Pulse-Width Modulator

The pulse-width modulation submodule has two output pins. The outputs are periodic waveforms controlled by a single frequency whose duty cycles may be independently selected and modified by user software. Each PWM can be independently programmed to run in fast or slow mode. The PWM unit has its own 16-bit free-running counter which is clocked by an output of the nine-stage prescaler (the same prescaler used by the capture/compare unit) or by the clock input pin, PCLK. Refer to the following block diagram of the PWM submodule.



**PWM Unit Block Diagram**

## 4.4 GPT Registers

**GPTMCR** — GPT Module Configuration Register **$YFF900**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STOP | FRZ | | STOPP | INCP | 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 |

The GPTMCR contains parameters for interfacing to the CPU and the intermodule bus.

STOP — Stop Clocks
0 = Internal clocks not shut down
1 = Internal clocks shut down

FRZ1 — FREEZE Response
Reserved; has no effect

FRZ0 — FREEZE Response
0 = Ignore FREEZE
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
0 = Registers with access controlled by SUPV are unrestricted (FC2 is a don't care).
1 = Registers with access controlled by SUPV are restricted when FC2 = 1.
Because the CPU16 operates in supervisor mode only (FC2 is always logic level one), this bit has no effect.

IARB[3:0] — Interrupt Arbitration ID
The value in this field is used to arbitrate between simultaneous interrupt service requests of the same priority. Each module that can generate interrupts has an IARB field — in order to implement an arbitration scheme, each IARB field must be set to a different non-zero value. If an interrupt request from a module that has an IARB field value of $0 is recognized, the CPU16 processes a spurious interrupt exception. The reset value of all IARB fields other than that of the SCIM is $0 (no priority), to preclude interrupt processing during reset.

**MTR** — GPT Module Test Register (Reserved) **$YFF902**

This address is currently unused and will return zeros if read. It is reserved for GPT factory test.

**ICR** — GPT Interrupt Configuration Register **$YFF904**

| 15 | | | 12 | 11 | 10 | | 8 | 7 | | | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PA | | | | 0 | IRL | | | VBA | | | | 0 | 0 | 0 | 0 |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

PA — Priority Adjust Field

This field specifies an interrupt to be advanced to the highest priority.

IRL— Interrupt Request Level

This field specifies the priority level of interrupts generated by the GPT.

VBA — Vector Base Address

This is the most significant nibble of interrupt vectors generated by the GPT.

**DDRGP/PORTGP** — Port GP Data Direction Register/Port GP Data Register **$YFF906**

| 15 | | | | | | | 8 | 7 | | | | | | | 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] — Port GP Data Direction Register

0 = Input only

1 = Output

When PORTGP is used for general-purpose I/O, each bit in DDRGP determines whether the corresponding PORTGP bit is input or output.

**OC1M/OC1D** — OC1 Action Mask Register/OC1 Action Data Register **$YFF908**

| 15 | | | | 11 | 10 | 9 | 8 | 7 | | | | 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 Field

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 Field

0 = If OC1 mask bit is set, clear the corresponding output compare pin on OC1 match.

1 = If OC1 mask bit is set, set the corresponding output compare pin on OC1 match.

OC1D[5:1] correspond to OC[5:1].

**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.

**PACTL/PACNT** — Pulse Accumulator Control Register/Counter **$YFF90C**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PAIS | PAEN | PAMOD | PEDGE | PCLKS | I4/O5 | PACLK | | PACNT | | | | | | | |
| 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 System 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.

### TIC1–TIC3 — Input Capture Registers 1–3 $YFF90E, $YFF910, $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.

### TOC1–TOC4 — Output Compare Registers 1–4 $YFF914, $YFF916, $YFF918, $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.

### 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.

### TCTL1/TCTL2 — Timer Control Registers 1–2 $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 Bits

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 |

| 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 — Capture/Compare 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] | Prescaler Value |
|---|---|
| 000 | 4 |
| 001 | 8 |
| 010 | 16 |
| 011 | 32 |
| 100 | 64 |
| 101 | 128 |
| 110 | 256 |
| 111 | PCLK |

**TFLG1/TFLG2** — Timer Interrupt Flag Registers 1–2 **$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.

**CFORC/PWMC** — Compare Force Register/PWM Control Register **$YFF924**

| | 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 |

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 PWMA

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.

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 on PWMA

0 = Force logic level zero output on PWMA pin.

1 = Force logic level one output on PWMA pin.

F1B — Force Logic Level on PWMB

0 = Force logic level zero output on PWMB pin.

1 = Force logic level one output on PWMB pin.

**PWMA/PWMB** — PWM Registers A/B **$YFF926, $YFF927**

These registers are associated with the pulse-width value of the PWM output on the corresponding PWM pin. A value of $00 loaded into one of these registers results in a continuously low output on the corresponding pin. A value of $80 results in a 50% duty cycle output. Maximum value ($FF) selects an output which is high for 255/256 of the period.

**PWMCNT** — PWM Count Register **$YFF928**

PWMCNT is the 16-bit free-running counter associated with the PWM functions of the GPT module.

**PWMBUFA/B** — PWM Buffer Registers A/B **$YFF92A, $YFF92B**

These read-only registers contain values associated with the duty cycles of the corresponding PWM. Reset state is $0000.

**PRESCL** — GPT Prescaler **$YFF92C**

The 9-bit prescaler value can be read from bits [8:0] at this address. Bits [15:9] will always read as zeros. Reset state is $0000.

## 5 Analog-to-Digital Converter Module

The 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. A block diagram of the ADC module follows.

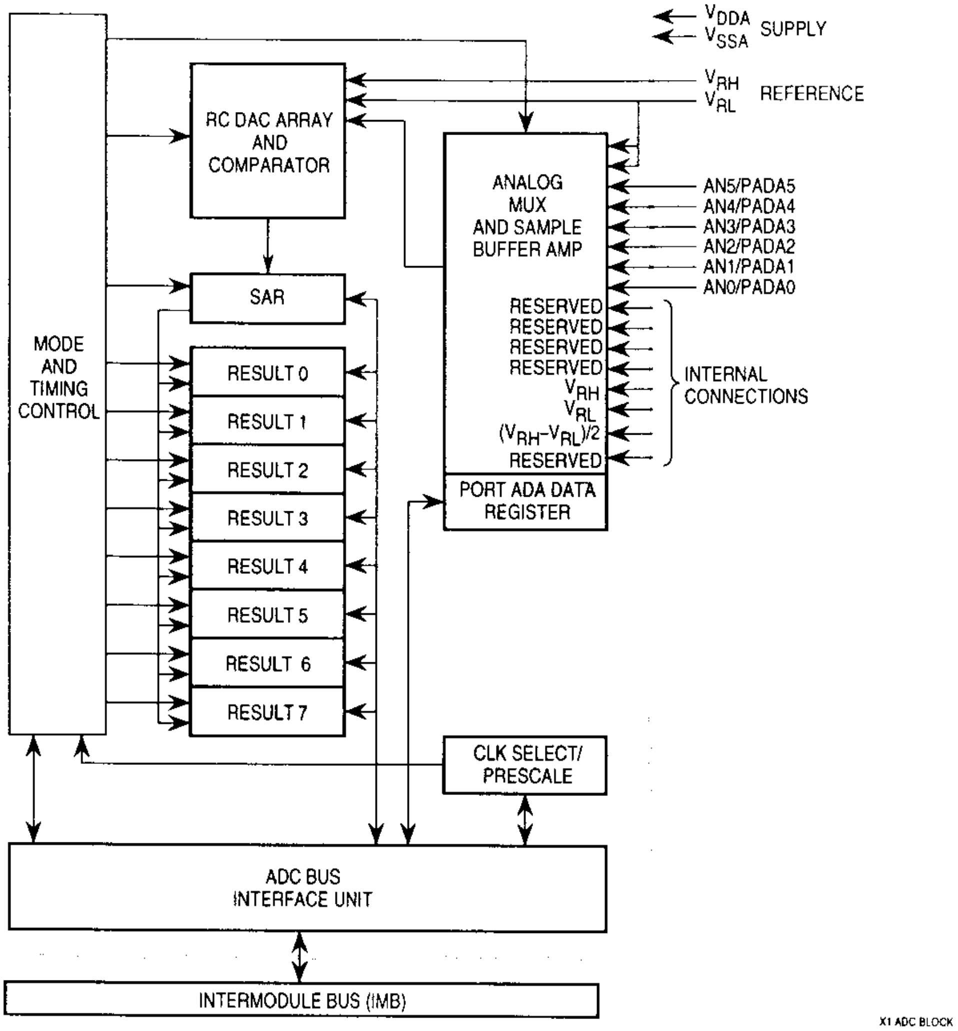
### ADC Module Address Map

| 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) | |

Y = M111, where M is the state of the modmap bit in SCIMCR. In an M68HC16 MCU, M must always be set to one.

## 5.1 Overview

ADC module conversion functions can be grouped into three basic subsystems: an analog front end, a digital control section, and result storage. In addition to use as multiplexer inputs, the eight analog inputs can be used as a general-purpose digital input port (port ADA), provided signals are within logic level specification.



**Analog-to-Digital Converter Block Diagram**

## 5.2 Analog Subsystem

The analog front end consists of a multiplexer, a buffer amplifier, a resistor-capacitor array, and a high-gain comparator. The multiplexer selects one of eight internal or eight external signal sources for conversion. The buffer amplifier protects the input channel from the relatively large capacitance of the resistor capacitor (RC) array. The RC array performs two functions. It acts as a sample/hold circuit, and it provides the digital-to-analog comparison output necessary for successive approximation conversion. The comparator indicates whether each successive output of the RC array is higher or lower than the sampled input.

### 5.3 Digital Control Subsystem

The digital control section includes conversion sequence control logic, channel and reference select logic, successive approximation register, eight result registers, a port data register, and control/status registers. It controls the multiplexer and the output of the RC array during the sample and conversion periods, stores the results of comparison in the successive-approximation register, then transfers the result to a result register.

### 5.4 Bus Interface Subsystem

The bus interface contains logic necessary to interface the ADC to the intermodule bus. The ADC is designed to act as a slave device on the bus. The interface must respond with appropriate bus cycle termination signals and supply appropriate interface timing to the other submodules.

### 5.5 ADC Registers

**ADCMCR** — Module Configuration Register **$YFF700**

| 15 | 14 | 13 | 12 | 8 | 7 | 6 | 0 |
|---|---|---|---|---|---|---|---|
| STOP | FRZ | | NOT USED | | SUPV | NOT USED | |
| RESET: | | | | | | | |
| 1 | 0 | 0 | | | 1 | | |

The module configuration register is used to initialize the ADC.

STOP — STOP Mode

0 = Normal operation

1 = Low-power operation

STOP places the ADC in low-power state by disabling the ADC clock and powering down the analog circuitry. Setting STOP will abort any conversion in progress. STOP is set to logic level one at reset, and may be cleared to logic level zero by the CPU.

Clearing STOP enables normal ADC operation. However, because analog circuitry bias current has been turned off, there is a period of recovery before output stabilization.

FRZ[1:0] — Freeze 1

The FRZ field is used to determine ADC response to assertion of the IFREEZE signal. The following table shows possible responses.

| FRZ | Response |
|---|---|
| 00 | Ignore IFREEZE |
| 01 | Reserved |
| 10 | Finish conversion, then freeze |
| 11 | Freeze immediately |

SUPV — Supervisor/Unrestricted

0 = Unrestricted access

1 = Supervisor access

SUPV defines access to assignable ADC registers. Because the CPU16 operates in supervisor mode only, this bit has no effect.

### ADTEST — ADC Test Register $YFF702

ADTEST is used with the SCIM test register for factory test of the ADC.

### PORTADA — Port Data Register $YFF706

| | 15 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| | NOT USED | | PADA7 | PADA6 | PADA5 | PADA4 | PADA3 | PADA2 | PADA1 | PADA0 |
| RESET: | | | | | | | | | | |
| | | | | | | INPUT DATA | | | | |

Port ADA is an input port that shares pins with the A/D converter inputs.

PADA[7:6]

Internally connected to $V_{RLP}$. The general-purpose input value is equal to zero.

PADA[5:0]

A read of PORTADA[5:0] returns the logic level of port AD pins [5:0]. If an input is not at an appropriate logic level (i.e., outside the defined levels), the read is indeterminate. Use of a port AD pin for digital input does not preclude use as an analog input.

### ADCTL0 — A/D Control Register 0 $YFF70A

| | 15 | 8 | 7 | 6 | 5 | 4 | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| | NOT USED | | RES10 | STS | | PRS | | | | |
| RESET: | | | | | | | | | | |
| | | | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

ADCTL0 is used to select ADC clock source and to set up prescaling. Writes to it have immediate effect.

RES10 — 10-Bit Resolution

0 = 8-bit conversion

1 = 10-bit conversion

Conversion results are appropriately aligned in result registers to reflect conversion status.

STS[1:0] — Sample Time Select Field

The STS field is used to select one of four sample times, as shown in the following table.

| STS[1:0] | Sample Time |
|---|---|
| 00 | 2 A/D Clock Periods |
| 01 | 4 A/D Clock Periods |
| 10 | 8 A/D Clock Periods |
| 11 | 16 A/D Clock Periods |

PRS[4:0] — Prescaler Rate Selection Field

ADC clock is generated from system clock using a modulo counter and a divide-by-two circuit. The binary value of this field is the counter modulus. System clock is divided by the PRS value plus one, then sent to the divide-by-two circuit, as shown in the following table. Maximum ADC clock rate is 2 MHz. Reset value of PRS is a divisor value of eight. This translates into a nominal 2 MHz ADC clock.

| PRS[4:0] | Divisor Value |
|---|---|
| 00000 | Reserved |
| 00001 | 4 |
| 00010 | 6 |
| ... | ... |
| 11101 | 60 |
| 11110 | 62 |
| 11111 | 64 |

**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 |

ADCTL1 is used to initiate A/D conversion. It is also used to select conversion modes and conversion channel. It can be written or read at any time. A write to ADCTL1 initiates a conversion sequence. If a conversion sequence is already in progress, a write to ADCTL1 will abort it and reset the SCF and CCF flags in the A/D status register.

SCAN — Scan Mode Selection Bit

0 = Single conversion sequence
1 = Continuous conversion

Length of conversion sequence(s) is determined by S8CM.

MULT — Multichannel Conversion Bit

0 = Conversion sequence(s) run on single channel (channel selected via [CD:CA])
1 = Sequential conversion of a block of four or eight channels (block selected via [CD:CA])

Length of conversion sequence(s) is determined by S8CM.

S8CM — Select Eight-Conversion Sequence Mode

0 = Four-conversion sequence
1 = Eight-conversion sequence

This bit determines the number of conversions in a conversion sequence.

[CD:CA] — Channel Selection Field

The bits in this field are used to select an input or block of inputs for A/D conversion.

The following table is a summary of the operation of S8CM and [CD:CA] when MULT is cleared (single-channel mode). Number of conversions per channel is determined by SCAN.

| S8CM | CD | CC | CB | CA | Input | Result Register |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | AN0 | RSLT0 – RSLT3 |
| 0 | 0 | 0 | 0 | 1 | AN1 | RSLT0 – RSLT3 |
| 0 | 0 | 0 | 1 | 0 | AN2 | RSLT0 – RSLT3 |
| 0 | 0 | 0 | 1 | 1 | AN3 | RSLT0 – RSLT3 |
| 0 | 0 | 1 | 0 | 0 | AN4 | RSLT0 – RSLT3 |
| 0 | 0 | 1 | 0 | 1 | AN5 | RSLT0 – RSLT3 |
| 0 | 0 | 1 | 1 | 0 | AN6* | RSLT0 – RSLT3* |
| 0 | 0 | 1 | 1 | 1 | AN7* | RSLT0 – RSLT3* |
| 0 | 1 | 0 | 0 | 0 | Reserved | RSLT0 – RSLT3 |
| 0 | 1 | 0 | 0 | 1 | Reserved | RSLT0 – RSLT3 |
| 0 | 1 | 0 | 1 | 0 | Reserved | RSLT0 – RSLT3 |
| 0 | 1 | 0 | 1 | 1 | Reserved | RSLT0 – RSLT3 |
| 0 | 1 | 1 | 0 | 0 | $V_{RH}$ | RSLT0 – RSLT3 |
| 0 | 1 | 1 | 0 | 1 | $V_{RL}$ | RSLT0 – RSLT3 |
| 0 | 1 | 1 | 1 | 0 | $(V_{RH} - V_{RL}) / 2$ | RSLT0 – RSLT3 |
| 0 | 1 | 1 | 1 | 1 | Test/Reserved | RSLT0 – RSLT3 |
| 1 | 0 | 0 | 0 | 0 | AN0 | RSLT0 – RSLT7 |
| 1 | 0 | 0 | 0 | 1 | AN1 | RSLT0 – RSLT7 |
| 1 | 0 | 0 | 1 | 0 | AN2 | RSLT0 – RSLT7 |
| 1 | 0 | 0 | 1 | 1 | AN3 | RSLT0 – RSLT7 |
| 1 | 0 | 1 | 0 | 0 | AN4 | RSLT0 – RSLT7 |
| 1 | 0 | 1 | 0 | 1 | AN5 | RSLT0 – RSLT7 |
| 1 | 0 | 1 | 1 | 0 | AN6* | RSLT0 – RSLT7* |
| 1 | 0 | 1 | 1 | 1 | AN7* | RSLT0 – RSLT7* |
| 1 | 1 | 0 | 0 | 0 | Reserved | RSLT0 – RSLT7 |
| 1 | 1 | 0 | 0 | 1 | Reserved | RSLT0 – RSLT7 |
| 1 | 1 | 0 | 1 | 0 | Reserved | RSLT0 – RSLT7 |
| 1 | 1 | 0 | 1 | 1 | Reserved | RSLT0 – RSLT7 |
| 1 | 1 | 1 | 0 | 0 | $V_{RH}$ | RSLT0 – RSLT7 |
| 1 | 1 | 1 | 0 | 1 | $V_{RL}$ | RSLT0 – RSLT7 |
| 1 | 1 | 1 | 1 | 0 | $(V_{RH} - V_{RL}) / 2$ | RSLT0 – RSLT7 |
| 1 | 1 | 1 | 1 | 1 | Test/Reserved | RSLT0 – RSLT7 |

*AN6 and AN7 are internally connected to $V_{RLP}$. Corresponding general-purpose input values are zero; result register value is $0000.

The following table is a summary of the operation of S8CM and [CD:CA] when MULT is set (multichannel mode). Number of conversions per channel is determined by SCAN. Channel numbers are given in order of conversion.

| 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 |

*AN6 and AN7 are internally connected to $V_{RLP}$. Corresponding general-purpose input values are zero; result register value is $0000.

**ADSTAT** — ADC Status Register **$YFF70E**

| 15 | 14 | | | 11 | 10 | | 8 | 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SCF | NOT USED | | | | CCTR | | | CCF | | | | | | | |
| RESET: | | | | | | | | | | | | | | | |
| 0 | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

ADSTAT contains information related to the status of a conversion sequence.

SCF — Sequence Complete Flag
0 = Sequence not complete
1 = Sequence complete

SCF is set at the end of the conversion sequence when SCAN is cleared, and at the end of the **first** conversion sequence when SCAN is set. SCF is cleared when ADCTL1 is written and a new conversion sequence begins.

CCTR[2:0] — Conversion Counter Field

This field reflects the contents of the conversion counter pointer in either four or eight count conversion sequence. The value corresponds to the number of the next result register to be written, and thus indicates which channel is being converted.

CCF[7:0] — Conversion Complete Field

Each bit in this field corresponds to an A/D result register (CCF7 to RSLT7, etc.). A bit is set when conversion for the corresponding channel is complete, and remains set until the result register is read. It is cleared when the register is read.

**RSLT0–RSLT7** — A/D Result Registers **$YFF710–$YFF73E**

The result registers store data after conversion is complete. Each register can be read from three different addresses in the register block. Data format depends on the address from which it is read.

**RJURR** — Unsigned Right-Justified Format **$YFF710–$YFF71F**

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

**LJSRR** — Signed Left-Justified Format **$YFF720–$YFF72F**

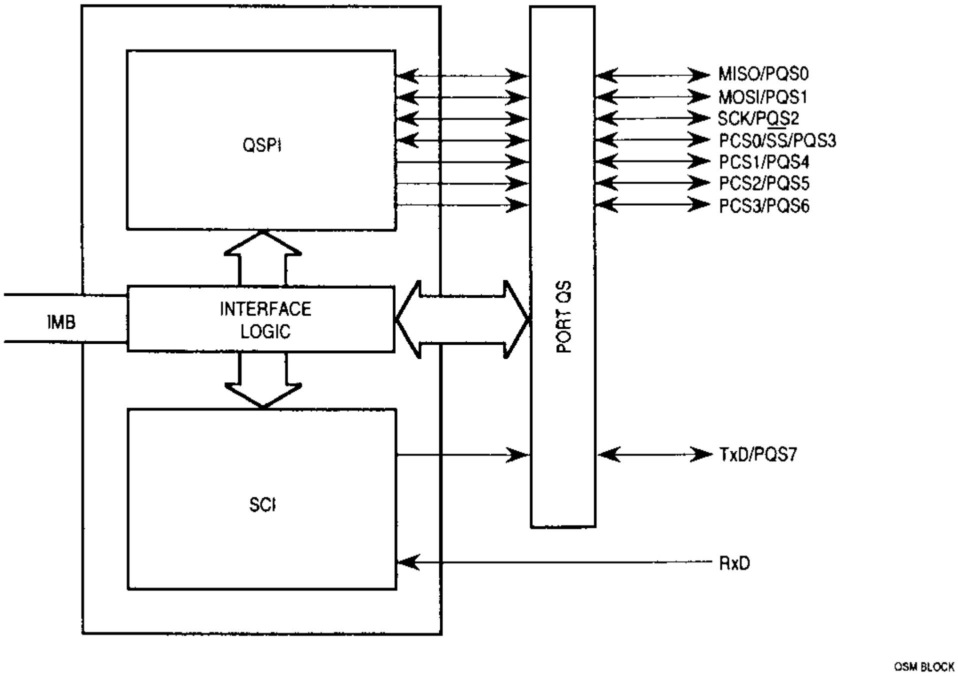
Conversion result is signed left-justified data. Bits [15:6] are used for 10-bit conversion, bits [15:8] are used for 8-bit conversion (bits [7:6] are zero). Although the ADC is unipolar, it is assumed that the zero point is halfway between low and high reference when this format is used. For positive input, bit 15 = 0, for negative input, bit 15 = 1. Bits [5:0] always return zero when read.

**LJURR** — Unsigned Left-Justified Format **$YFF730–$YFF73F**

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

# 6 Queued Serial Module

The QSM contains two serial interfaces, the queued serial peripheral interface (QSPI) and the serial communication interface (SCI). An address map of the QSM follows.



**QSM Block Diagram**

## 6.1 Overview

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

The SCI provides a standard nonreturn to zero (NRZ) mark/space format. It operates in either full- or half-duplex mode. There are separate transmitter and receiver enable bits and dual data buffers. A modulus-type baud rate generator provides rates from 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.

**QSM Address Map**

| Address | 15 8 | 7 0 |
|---|---|---|
| $YFFC00 | QSM MODULE CONFIGURATION (QSMCR) | |
| $YFFC02 | QSM TEST (QTEST) | |
| $YFFC04 | QSM INTERRUPT LEVEL (QILR) | 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) |
| $YFFC20–<br>$YFFCFF | RESERVED | |
| $YFFD00–<br>$YFFD1F | RECEIVE RAM (RR[0:F]) | |
| $YFFD20–<br>$YFFD3F | TRANSMIT RAM (TR[0:F]) | |
| $YFFD40–<br>$YFFD4F | COMMAND RAM (CR[0:F]) | |

Y = M111, where M is the state of the modmap bit in SCIMCR. In an M68HC16 MCU, M must always be set to one.

## 6.2 QSM Registers

There are four types of QSM registers: QSM global registers, QSM pin control registers, QSPI submodule registers, and SCI submodule registers. The QSPI and SCI registers are defined in separate sections below. Writes to unimplemented register bits have no meaning or effect, and reads from unimplemented bits always return a logic zero value.

### 6.2.1 Global Registers

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

**QSMCR** — QSM Configuration Register **$YFFC00**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 |  |  | 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 |

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

STOP — Stop Enable

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

STOP places the QSM in a low-power state by disabling the system clock in most parts of the module. QSMCR is the only register guaranteed to be readable while STOP is asserted. The QSPI RAM is not readable. However, writes to RAM or any register are guaranteed to be valid while STOP is asserted. STOP can be negated by the CPU and by reset.

The system software must stop each submodule before asserting STOP to avoid complications at restart and to avoid data corruption. The SCI submodule receiver and transmitter should be disabled, and the operation should be verified for completion before asserting STOP. The QSPI submodule should be stopped by asserting the HALT bit in SPCR3 and by asserting STOP after the HALTA flag is set.

FRZ1 — Freeze 1

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

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

FRZ0 — Freeze 0

Reserved

SUPV — Supervisor/Unrestricted

0 = User access
1 = Supervisor access

SUPV defines the assignable QSM registers as either supervisor-only data space or unrestricted data space. Because the CPU16 operates in supervisor mode only, this bit has no effect.

IARB — Interrupt Arbitration Identification Number

The value in this field is used to arbitrate between simultaneous interrupt service requests of the same priority. Each module that can generate interrupts has an IARB field — in order to implement an arbitration scheme, each IARB field must be set to a different non-zero value. If an interrupt request from a module that has an IARB field value of $0 is recognized, the CPU16 processes a spurious interrupt exception. The reset value of all IARB fields other than that of the SCIM is $0 (no priority), to preclude interrupt processing during reset.

**QTEST** — QSM Test Register **$YFFC02**

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

**QILR** — QSM Interrupt Levels Register **$YFFC04**

| 15 | 14 | 13 |  | 11 | 10 |  | 8 | 7 |  |  |  |  |  |  | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | ILQSPI |  |  | ILSCI |  |  | QIVR |  |  |  |  |  |  |  |
| RESET: |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |

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

ILQSPI — Interrupt Level for QSPI

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

ILSCI — Interrupt Level of SCI

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

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

**QIVR** — QSM Interrupt Vector Register **$YFFC05**

| 15 |  |  |  |  |  |  | 8 | 7 |  |  |  |  |  |  | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QILR |  |  |  |  |  |  |  | INTV |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | RESET: |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

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

After initialization, QIVR determines which two vectors in the exception vector table are to be used for QSM interrupts. The QSPI and SCI submodules have separate interrupt vectors adjacent to each other. Both submodules use the same interrupt vector with the least significant bit (LSB) determined by the submodule causing the interrupt.

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

### 6.2.2 Pin Control Registers

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

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

**PORTQS** — Port QS Data Register **$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 is the port QS data register. Writes to PORTQS affect pins defined as outputs. Reads of PORTQS return data present on the pins.

**PQSPAR** — Port QS Pin Assignment Register **$YFFC16**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7–0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PQSPA6 | PQSPA5 | PQSPA4 | PQSPA3 | 0 | PQSPA1 | PQSPA0 | DDRQS |
| RESET: |  |  |  |  |  |  |  |  |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |

PQSPAR determines whether certain pins are used by the QSPI submodule, or whether they are available for general-purpose I/O. Pins designated for general-purpose I/O are controlled by DDRQS and PORTQS. PQSPAR does not affect operation of the SCI submodule. Bits 15 and 10 are not implemented.

**PQSPAR Pin Assignments**

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

NOTES:

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

## **DDRQS** — Port QS Data Direction Register **$YFFC17**

| 15 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| PQSPAR | | DDQS7 | DDQS6 | DDQS5 | DDQS4 | DDQS3 | DDQS2 | DDQS1 | DDQS0 |
| | | RESET: | | | | | | | |
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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.

### Effect of DDRQS on PORTQS Pins

| Pin | DDRQS Bit | Pin Function |
|---|---|---|
| PQS0 | 0 | Digital Input |
| | 1 | Digital Output |
| PQS1 | 0 | Digital Input |
| | 1 | Digital Output |
| PQS2 | 0 | Digital Input |
| | 1 | Digital Output |
| PQS2 | 0 | Digital Input |
| | 1 | Digital Output |
| PQS3 | 0 | Digital Input |
| | 1 | Digital Output |
| PQS4 | 0 | Digital Input |
| | 1 | Digital Output |
| PQS5 | 0 | Digital Input |
| | 1 | Digital Output |
| PQS6 | 0 | Digital Input |
| | 1 | Digital Output |
| PQS7 | 0 | Digital Input |
| | 1 | Digital Output |

### Effect of DDRQS on QSM Pin Function

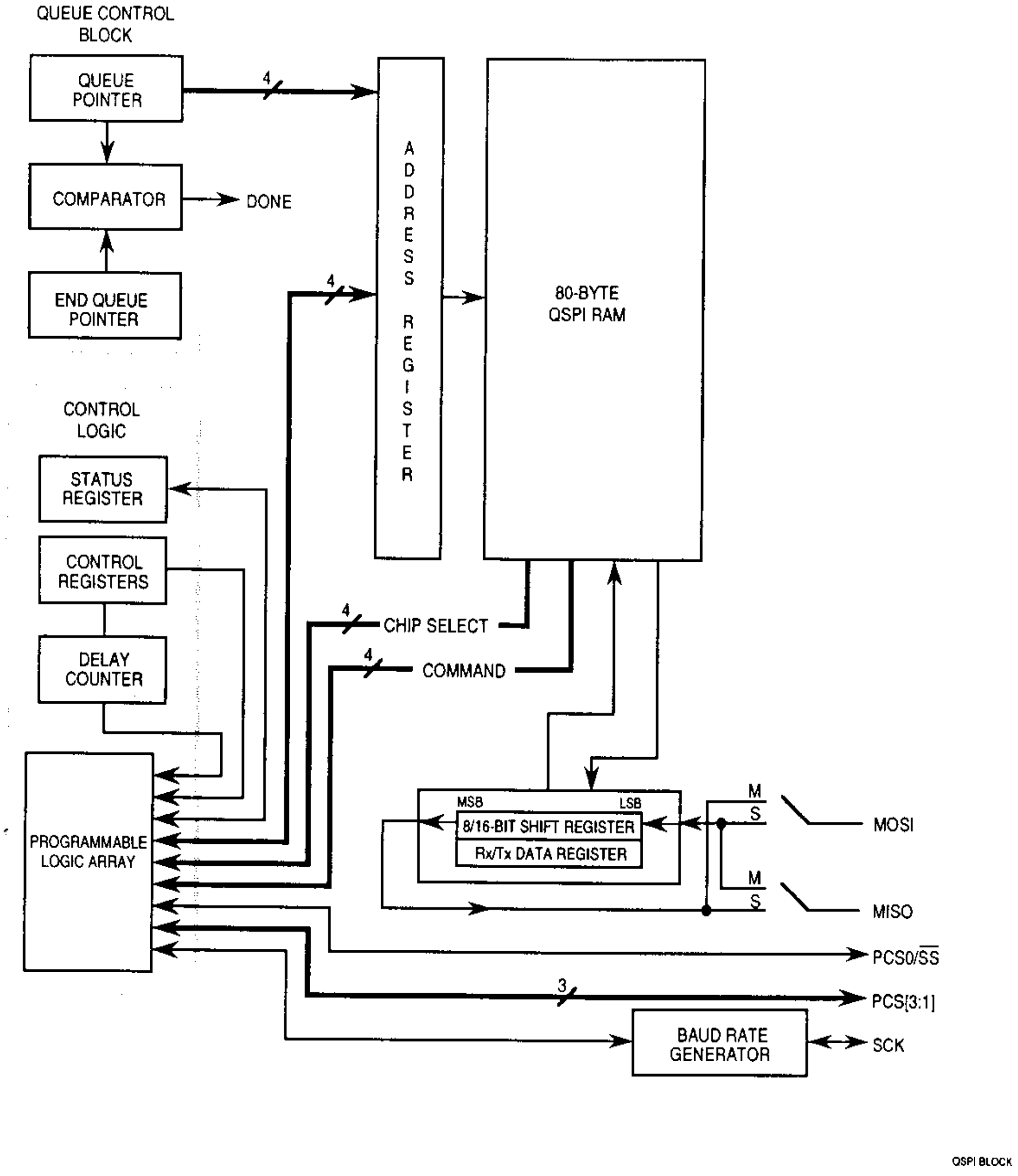
| QSM Pin | Mode | DDRQS Bit | Bit State | Pin Function |
|---|---|---|---|---|
| MISO | Master | DDQS0 | 0 | Serial Data Input to QSPI |
| | | | 1 | Disables Data Input |
| | Slave | | 0 | Disables Data Output |
| | | | 1 | Serial Data Output from QSPI |
| MOSI | Master | DDQS1 | 0 | Disables Data Output |
| | | | 1 | Serial Data Output from QSPI |
| | Slave | | 0 | Serial Data Input to QSPI |
| | | | 1 | Disables Data Input |
| SCK[1] | Master | DDQS2 | 0 | Disables Clock Output |
| | | | 1 | Clock Output from QSPI |
| | Slave | | 0 | Clock Input to QSPI |
| | | | 1 | Disables Clock Input |
| PCS0/$\overline{SS}$ | Master | DDQS3 | 0 | Assertion Causes Mode Fault |
| | | | 1 | Chip-Select Output |
| | Slave | | 0 | QSPI Slave Select Input |
| | | | 1 | Disables Select Input |
| PCS[3:1] | Master | DDQS [4:6] | 0 | Disables Chip-Select Output |
| | | | 1 | Chip-Select Output |
| | Slave | | 0 | Inactive |
| | | | 1 | Inactive |
| TXD[2] | Transmit | DDQS7 | X | Serial Data Output from SCI |
| RXD | Receive | None | NA | Serial Data Input to SCI |

NOTES:

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

## 6.3 QSPI Submodule

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



QSPI Block Diagram

### 6.3.1 QSPI Pins

Seven pins are associated with the QSPI. When not needed for a QSPI application, they can be configured as general-purpose I/O pins.

Refer to the following table for QSPI input and output pins and their functions.

| Pin Names | Mnemonics | Mode | Function |
|---|---|---|---|
| Master In Slave Out | MISO | Master<br>Slave | Serial Data Input to QSPI<br>Serial Data Output from QSPI |
| Master Out Slave In | MOSI | Master<br>Slave | Serial Data Output from QSPI<br>Serial Data Input to QSPI |
| Serial Clock | SCK | Master<br>Slave | Clock Output from QSPI<br>Clock Input to QSPI |
| Peripheral Chip Selects | PCS[3:0] | Master | Select Peripherals |
| Slave Select | $\overline{\text{SS}}$ | Master<br>Slave | Causes Mode Fault<br>Initiates Serial Transfer |

### 6.3.2 QSPI Registers

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

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

Refer to the following memory map of the QSPI.

| Address | Name | Usage |
|---|---|---|
| $YFFC18 | SPCR0 | QSPI Control Register 0 |
| $YFFC1A | SPCR1 | QSPI Control Register 1 |
| $YFFC1C | SPCR2 | QSPI Control Register 2 |
| $YFFC1E | SPCR3 | QSPI Control Register 3 |
| $YFFC1F | SPSR | QSPI Status Register |
| $YFFD00 | RAM | QSPI Receive Data (16 Words) |
| $YFFD20 | RAM | QSPI Transmit Data (16 Words) |
| $YFFD40 | RAM | QSPI Command Control<br>(8 Words) |

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

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

| 15 | 14 | 13 | | | 10 | 9 | 8 | 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MSTR | WOMQ | BITS | | | | CPOL | CPHA | SPBR | | | | | | | |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

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

MSTR — Master/Slave Mode Select

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

MSTR configures the QSPI for either master or slave mode operation. This bit is cleared on reset and may only be written by the CPU.

WOMQ — Wired-OR Mode for QSPI Pins

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

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

BITS — Bits Per Transfer

In master mode, when BITSE in a command is set, the BITS field determines the number of data bits transferred. When BITSE is cleared, 8 bits are transferred. Reserved values default to 8 bits. BITSE is not used in slave mode.

The following table shows the number of bits per transfer.

| 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 |

CPOL — Clock Polarity

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

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

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

CPHA — Clock Phase

0 = Data is captured on the leading edge of SCK and changed on the following edge of SCK.

1 = Data is changed on the leading edge of SCK and captured on the following edge of SCK.

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

SPBR — Serial Clock Baud Rate

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

$$\text{SCK BAUD RATE} = \frac{\text{SYSTEM CLOCK}}{\text{2SPBR}}$$

or

$$\text{SPBR} = \frac{\text{SYSTEM CLOCK}}{\text{(2SCK) (BAUD RATE DESIRED)}}$$

where SPBR equals {2, 3, 4,..., 255}

Giving SPBR a value of zero or one disables the baud rate generator. SCK is disabled and assumes its inactive state value. No serial transfers occur. At reset, SPBR is initialized to a 2.1-MHz SCK frequency.

**SPCR1** — QSPI Control Register 1 **$YFFC1A**

| 15 | 14 | | | | | | 8 | 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SPE | DSCKL | | | | | | | DTL | | | | | | | |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

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

SPE — QSPI Enable

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

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

DSCKL — Delay before SCK

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

$$\text{PCS to SCK DELAY} = \frac{\text{DSCKL}}{\text{SYSTEM CLOCK}}$$

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

When a queue entry's DSCK equals zero, then DSCKL is not used. Instead, the PCS valid-to-SCK transition is one-half SCK period.

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. The following equation is used to calculate the delay:

$$\text{DELAY AFTER TRANSFER} = \frac{\text{32DTL}}{\text{SYSTEM CLOCK}}$$

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

A zero value for DTL causes the following delay-after-transfer value:

$$\frac{8192}{\text{SYSTEM CLOCK}}$$

If DT equals zero, a standard delay is inserted.

$$\text{STANDARD DELAY AFTER TRANSFER} = \frac{17}{\text{SYSTEM CLOCK}}$$

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

**SPCR2** — QSPI Control Register 2 **$YFFC1C**

| 15 | 14 | 13 | 12 | 11 | | | 8 | 7 | 6 | 5 | 4 | 3 | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SPIFIE | WREN | WRTO | 0 | ENDQP | | | | 0 | 0 | 0 | 0 | NEWQP | | | |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

SPIFIE — SPI Finished Interrupt Enable

0 = QSPI interrupts disabled
1 = QSPI interrupts enabled

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

WREN — Wrap Enable

0 = Wraparound mode disabled
1 = Wraparound mode enabled

WREN enables or disables wraparound mode.

WRTO — Wrap To

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

ENDQP — Ending Queue Pointer

This field contains the last QSPI queue address.

NEWQP — New Queue Pointer Value

This field contains the first QSPI queue address.

**SPCR3** — QSPI Control Register 3 **$YFFC1E**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | LOOPQ | HMIE | HALT | SPSR | |
| RESET: | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |

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

LOOPQ — QSPI Loop Mode

0 = Feedback path disabled
1 = Feedback path enabled

LOOPQ controls feedback on the data serializer for testing.

HMIE — HALTA and MODF Interrupt Enable

0 = HALTA and MODF interrupts disabled
1 = HALTA and MODF interrupts enabled

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

HALT — Halt

0 = Halt not enabled
1 = Halt enabled

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

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

| 15 | 8 | 7 | 6 | 5 | 4 | 3 | | | 0 |
|---|---|---|---|---|---|---|---|---|---|
| SPCR3 | | SPIF | MODF | HALTA | 0 | CPTQP | | | |
| | RESET: | | | | | | | | |
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SPSR contains QSPI status information. Only the QSPI can assert the bits in this register. The CPU reads this register to obtain status information and writes it to clear status flags.

SPIF — QSPI Finished Flag
0 = QSPI not finished
1 = QSPI finished
SPIF is set after execution of the command at the address in ENDQP.

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

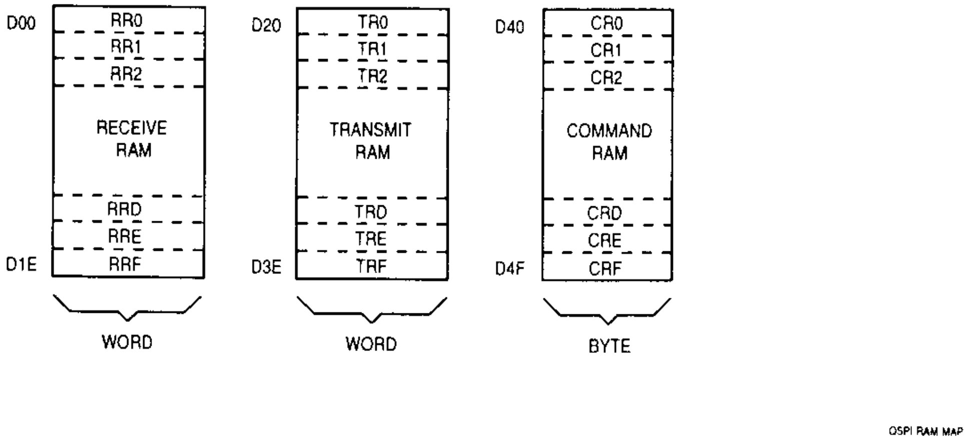
HALTA — Halt Acknowledge Flag
0 = QSPI not halted
1 = QSPI halted
HALTA is asserted when the QSPI halts in response to CPU assertion of HALT.

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.

### 6.3.3 QSPI RAM

The QSPI contains an 80-byte block of dual-access static RAM that is used by both the QSPI and the CPU. The RAM is divided into three segments: receive data RAM, transmit data RAM, and command control RAM. Receive data is information received from a serial device external to the MCU. Transmit data is information stored by the CPU for transmission to an external peripheral. Command control data is used to perform the transfer.

Refer to the following illustration of QSPI RAM organization.



**QSPI RAM Address Map**

Once the CPU has set up the queue of QSPI commands and enabled the QSPI, the QSPI can operate independently of the CPU. The QSPI executes all of the commands in its queue, sets a flag indicating that it is finished, and then either interrupts the CPU or waits for CPU intervention. It is possible to execute a queue of commands repeatedly without CPU intervention.

### RR[0:F] — Receive Data RAM $YFFD00

Data received by the QSPI is stored in this segment. The CPU reads this segment to retrieve data from the QSPI. Data stored in receive RAM is right-justified. Unused bits in a receive queue entry are set to zero by the QSPI upon completion of the individual queue entry. The CPU can access the data using byte, word, or long-word addressing.

The CPTQP value in SPSR shows which queue entries have been executed. The CPU uses this information to determine which locations in receive RAM contain valid data before reading them.

### TR[0:F] — Transmit Data RAM $YFFD20

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

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

**CR[0:F]** — Command RAM **$YFFD40**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CONT | BITSE | DT | DSCK | PCS3 | PCS2 | PCS1 | PCS0* |
| — | — | — | — | — | — | — | — |
| CONT | BITSE | DT | DSCK | PCS3 | PCS2 | PCS1 | PCS0* |
| COMMAND CONTROL | | | | PERIPHERAL CHIP SELECT | | | |

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

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

CONT — Continue

0 = Control of chip selects returned to PORTQS after transfer is complete.

1 = Peripheral chip selects remain asserted after transfer is complete.

BITSE — Bits per Transfer Enable

0 = 8 bits

1 = Number of bits set in BITS field of SPCR0

DT — Delay after Transfer

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

PCS[3:0] — Peripheral Chip Select

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

$\overline{\text{SS}}$ — Slave Mode Select

Initiates slave mode serial transfer. If $\overline{\text{SS}}$ is taken low when the QSPI is in master mode, a mode fault will be generated.

### 6.3.4 Operating Modes

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

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

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

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

## 6.4 SCI Submodule

The SCI submodule is used to communicate with external devices through an asynchronous serial bus. The SCI is fully compatible with the SCI systems found on other Motorola MCUs, such as the M68HC11 and M68HC05 Families.

### 6.4.1 SCI Pins

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

The following table shows SCI pins and their functions.

| Pin Names | Mnemonics | Mode | Function |
|---|---|---|---|
| Receive Data | RXD | Receiver Disabled<br>Receiver Enabled | Not Used<br>Serial Data Input to SCI |
| Transmit Data | TXD | Transmitter Disabled<br>Transmitter Enabled | General-Purpose I/O<br>Serial Data Output from SCI |

### 6.4.2 SCI Registers

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

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

**SCCR0** — SCI Control Register 0 **$YFFC08**

| | 15 | 14 | 13 | 12 | | | | | | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | SCBR | | | | | | | | | | | | |
| RESET: | | | | | | | | | | | | | | | | |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

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

SCBR — Baud Rate

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

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

$$\text{SCI BAUD RATE} = \frac{\text{SYSTEM CLOCK}}{32(\text{SCBR})}$$

or

$$\text{SCBR} = \text{SYSTEM CLOCK (32SCK) (BAUD RATE DESIRED)}$$

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

Writing a value of zero to SCBR disables the baud rate generator.

**SCCR1** — SCI Control Register 1 **$YFFC0A**

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | LOOPS | WOMS | ILT | PT | PE | M | WAKE | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| RESET: | | | | | | | | | | | | | | | | |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

LOOPS — Loop Mode

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

1 = Test SCI operation, looping, feedback path enabled

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

WOMS — Wired-OR Mode for SCI Pins

0 = If configured as an output, TXD is a normal CMOS output.
1 = If configured as an output, TXD is an open-drain output.

WOMS determines whether the TXD pin is an open-drain output or a normal CMOS output. This bit is used only when TXD is an output. If TXD is used as a general-purpose input pin, WOMS has no effect.

ILT — Idle-Line Detect Type

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

PT — Parity Type

0 = Even parity
1 = Odd parity

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

PE — Parity Enable

0 = SCI parity disabled
1 = SCI parity enabled

PE determines whether parity is enabled or disabled for both the receiver and the transmitter. If the received parity bit is not correct, the SCI sets the PF error flag in SCSR.

When PE is set, the most significant bit (MSB) of the data field is used for the parity function, which results in either seven or eight bits of user data, depending on the condition of M bit. The following table lists the available choices.

| M | PE | Result |
|---|---|---|
| 0 | 0 | 8 Data Bits |
| 0 | 1 | 7 Data Bits, 1 Parity Bit |
| 1 | 0 | 9 Data Bits |
| 1 | 1 | 8 Data Bits, 1 Parity Bit |

M — Mode Select

0 = SCI frame: 1 start bit, 8 data bits, 1 stop bit (10 bits total)
1 = SCI frame: 1 start bit, 9 data bits, 1 stop bit (11 bits total)

WAKE — Wakeup by Address Mark

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

TIE — Transmit Interrupt Enable

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

TCIE — Transmit Complete Interrupt Enable

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

RIE — Receiver Interrupt Enable

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

ILIE — Idle-Line Interrupt Enable

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

TE — Transmitter Enable

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

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

RE — Receiver Enable

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

RWU — Receiver Wakeup

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

Setting RWU enables the wakeup function, which allows the SCI to ignore received data until awakened by either an idle line or address mark (as determined by WAKE). When in wakeup mode, the receiver status flags are not set, and interrupts are inhibited. This bit is cleared automatically (returned to normal mode) when the receiver is awakened.

SBK — Send Break

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

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

**SCSR** — SCI Status Register **$YFFC0C**

| 15 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| NOT USED | | TDRE | TC | RDRF | RAF | IDLE | OR | NF | FE | PF |
| RESET: | | | | | | | | | | |
| | | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

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

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

TDRE — Transmit Data Register Empty Flag

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

1 = A new character can now be written to register TDR.

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

TC — Transmit Complete Flag

0 = SCI transmitter is busy

1 = SCI transmitter is idle

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

RDRF — Receive Data Register Full Flag

0 = Register RDR is empty or contains previously read data.

1 = Register RDR contains new data.

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

RAF — Receiver Active Flag

0 = SCI receiver is idle

1 = SCI receiver is busy

RAF indicates whether the SCI receiver is busy. It is set when the receiver detects a possible start bit and is cleared when the chosen type of idle line is detected. RAF can be used to reduce collisions in systems with multiple masters.

IDLE — Idle-Line Detected Flag

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

1 = SCI receiver detected an idle-line condition.

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

OR — Overrun Error Flag

0 = RDRF is cleared before new data arrives.

1 = RDRF is not cleared before new data arrives.

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

NF — Noise Error Flag

0 = No noise detected on the received data

1 = Noise occurred on the received data

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

FE — Framing Error Flag

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

0 = No framing error on the received data.

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

PF — Parity Error Flag

1 = Parity error occurred on the received data

0 = No parity error on the received data

PF is set when the SCI receiver detects a parity error. PF is not set until the entire frame is received and RDRF is set.

**SCDR** — SCI Data Register **$YFFC0E**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | R8/T8 | R7/T7 | R6/T6 | R5/T5 | R4/T4 | R3/T3 | R2/T2 | R1/T1 | R0/T0 |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | U | U | U | U | U | U | U | U | U |

SCDR contains two internal data registers at the same address. The receive data register (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. The transmit data register (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 Standby RAM Module

The standby RAM module (SRAM) provides 2 Kbytes of fast RAM that is especially useful for system stacks and variable storage. The SRAM has a dedicated power supply pin so that memory content can be preserved when the MCU is powered down.

## 7.1 Overview

The SRAM module consists of a control register block that is located at a fixed range of addresses in MCU address space, and a 2-Kbyte array of two bus cycle static RAM that can be mapped to any 2-Kbyte boundary in MCU address space. SRAM control registers are located at addresses $YFFB00–YFFB08, as shown in the address map.

The module responds to program and data space accesses. Data can be read or written in bytes, words, or long words. The RAM array must not be mapped so that array addresses overlap module control register addresses, as overlap makes the registers inaccessible.

SRAM is powered by $V_{DD}$ in normal operation. During power-down, SRAM contents are maintained by power from the $V_{STBY}$ input. Power switching between sources is automatic.

**SRAM Address Map**

| Address | 15 | 8 | 7 | 0 |
|---|---|---|---|---|
| $YFFB00 | SRAM MODULE CONFIGURATION REGISTER (RAMMCR) | | | |
| $YFFB02 | SRAM TEST REGISTER (RAMTST) | | | |
| $YFFB04 | SRAM ARRAY BASE ADDRESS REGISTER HIGH (RAMBAH) | | | |
| $YFFB06 | SRAM ARRAY BASE ADDRESS REGISTER LOW (RAMBAL) | | | |
| $YFFB08 | RESERVED | | | |

Y = M111, where M is the state of the modmap bit in SCIMCR. In an M68HC16 MCU, M must always be set to one.

## 7.2 SRAM Register Block

There are four SRAM control registers: the SRAM module configuration register (RAMMCR), the SRAM test register (RAMTST), and the SRAM array base address registers (RAMBAH/RAMBAL).

There is an 8-byte minimum register block size for the module. Unimplemented register addresses are read as zeros. Writes have no effect.

## 7.3 SRAM Registers

The CPU16 operates only in supervisory mode. Access to the SRAM array is controlled by the RASP field in RAMMCR. SRAM responds to both program and data space accesses based on the value in the RASP field in RAMMCR. This allows code to be executed from RAM, and permits the use of program counter relative addressing mode for operand fetches from the array.

**RAMMCR** — RAM Module Configuration Register **$YFFB00**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| STOP | 0 | 0 | 0 | RLCK | 0 | RASP | | NOT USED | |
| RESET: | | | | | | | | | |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | |

Use RAMMCR to determine whether the RAM is in STOP mode or normal mode. RAMMCR can determine in which space the array resides and also 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 controls whether the RAM array is in stop mode or normal operation. Reset state is one, leaving the array configured for LPSTOP operation. In stop mode, the array retains its contents, but cannot be read or written by the CPU. Because the CPU16 operates in supervisor mode, 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 to one once.

RASP[1:0] — RAM Array Space Field

This field 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 |

**RAMTST** — RAM Test Register **$YFFB02**

RAMTST is for factory test only. Reads of this register return zeros and writes have no effect.

**RAMBAH** — Array Base Address Register High **$YFFB04**

| 15–8 | 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 |

*ADDR[23:20] is at the same logic level as ADDR19 during internal CPU master operation. ADDR[23:20] must match ADDR19 for the chip select to be active.

**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 specify an SRAM base address in the system memory map. They can only be written while the SRAM is in low-power mode (RAMMCR STOP = 1, the default out of reset) and the base address lock is disabled (RAMMCR RLCK = 0, the default out of reset). This prevents accidental remapping of the array. Because the CPU16 drives ADDR[23:20] with the value of ADDR19, the value in the ADDR[23:20] fields must match the value in the ADDR19 field for the array to be accessible.

### 7.4 SRAM Operation

There are five operating modes.

a. The RAM module is in normal mode when powered by $V_{DD}$. The array can be accessed by byte, word, or long word. A byte or aligned word (high-order byte is at an even address) access only takes one bus cycle or two system clocks. A long word or misaligned word access requires two bus cycles.

b. Standby mode is intended to preserve RAM contents when $V_{DD}$ is removed. SRAM contents are maintained by a power source connected to the $V_{STBY}$ pin. The standby voltage is referred to as $V_{SB}$. Circuitry within the SRAM module switches to the higher of $V_{DD}$ or $V_{SB}$ with no loss of data. When SRAM is powered from the $V_{STBY}$ pin, access to the array is not guaranteed. If standby operation is not desired, connect the $V_{STBY}$ pin to $V_{SS}$.

c. Reset mode allows the CPU to complete the current bus cycle before resetting. 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.

d. Test mode is used for factory testing of the RAM array.

e. Writing the STOP bit of RAMMCR causes the SRAM module to enter stop mode. The RAM array is disabled which, if necessary, allows external logic to decode SRAM addresses but all data is retained. If $V_{DD}$ falls below $V_{SB}$, internal circuitry switches to $V_{SB}$, as in standby mode. Exit the stop mode by clearing the STOP bit.

# 8 Masked ROM Module

The masked ROM module (MRM) provides 48 Kbytes of nonvolatile, fast-access read-only system memory. The module can be configured to provide bootstrap vectors for system reset.

## 8.1 Overview

The MRM consists of a fixed-location control register block and a memory array. The primary function of the module is to serve as nonvolatile memory for the microcontroller. The CPU16 differentiates between program space accesses and data space accesses. The MRM array can be used for program code only, or for both program code and data. The MRM can also be programmed to insert wait states to accommodate migration from slower external development memory to the ROM array without retiming.

Configuration information is contained in the register block. Default reset base address of the array in the system address map is specified by the customer, but the array can be remapped to other addresses. In addition to the array base address, the register block contains operating parameters, bootstrap code, and ROM verification information. Refer to the following address map of the register block.

**MRM Control Register Address Map**

| Address | 15 8 | 7 0 |
|---|---|---|
| $YFF820 | MASKED ROM MODULE CONFIGURATION REGISTER (MRMCR) | |
| $YFF822 | NOT IMPLEMENTED | |
| $YFF824 | ARRAY BASE ADDRESS REGISTER HIGH (ROMBAH) | |
| $YFF826 | ARRAY BASE ADDRESS REGISTER LOW (ROMBAL) | |
| $YFF828 | ROM SIGNATURE HIGH REGISTER (RSIGHI) | |
| $YFF82A | ROM SIGNATURE LOW REGISTER (RSIGLO) | |
| $YFF82C | NOT IMPLEMENTED | |
| $YFF82E | NOT IMPLEMENTED | |
| $YFF830 | ROM BOOTSTRAP WORD 0 (ROMBS0) | |
| $YFF832 | ROM BOOTSTRAP WORD 1 (ROMBS1) | |
| $YFF834 | ROM BOOTSTRAP WORD 2 (ROMBS2) | |
| $YFF836 | ROM BOOTSTRAP WORD 3 (ROMBS3) | |
| $YFF838 | NOT IMPLEMENTED | |
| $YFF83A | NOT IMPLEMENTED | |
| $YFF83C | NOT IMPLEMENTED | |
| $YFF83E | NOT IMPLEMENTED | |

Y = M111, where M is the state of the modmap bit in SCIMCR. In an M68HC16 MCU, M must always be set to one.

The ROM array contains 48 Kbytes. It is arranged in 16-bit words, and is accessed via the intermodule bus. Bytes, words, and misaligned words can be accessed. Access time depends upon the number of wait states specified at mask programming time, but can be as fast as two system clocks for byte and aligned words. The MRM also responds to back-to-back IMB accesses to provide two bus cycle long word access.

The array base address must be on a 64-Kbyte boundary, must not overlap the control registers of other microcontroller modules, and should not overlap the control register block. The array occupies the low-order locations in the 64-Kbyte block. Accesses to the remaining 16 Kbytes of unimplemented locations in the block are ignored by the MRM, allowing other system resources or external devices to respond to the access. If the array is mapped to overlap the control registers of other modules, accesses to those registers are indeterminate; if the array is mapped to overlap the MRM control registers, accesses to the registers are still possible, but accesses to the overlapping 32 bytes of ROM are ignored.

The MRM can also operate in a special emulator mode that simplifies emulation of the array by an external device. Emulation mode is enabled by the EMUL bit in the MRMCR. EMUL state is determined by the state of the DATA10 and DATA13 lines during reset. If both data lines are held low, EMUL is set, and ROM emulation mode is enabled.

While emulation mode is enabled, the internal module chip select signal ($\overline{\text{CSM}}$) is asserted whenever a valid access to an address assigned to the masked ROM module is made. To be valid, an access must be within the range specified by the ROM base array registers and must meet the address space requirements defined by the ASPC field in MRMCR. $\overline{\text{CSM}}$ is asserted for all valid read accesses; it is asserted for write accesses only in background debug mode. The MRM does not acknowledge an access on the IMB while in emulation mode. This causes the SCIM to run an external bus cycle. The $\overline{\text{CSM}}$ signal is asserted on the falling edge of $\overline{\text{AS}}$. Internal $\overline{\text{DSACK}}$ is generated by the ROM module after it has inserted the number of wait states specified by the WAIT field in the MRMCR.

## 8.2 Masked ROM Control Registers

The 32-byte control register block contains registers that are used to configure the MRM and to control ROM array function. Configuration information is specified and programmed at the same time as the ROM content.

**MRMCR** — Masked ROM Module Configuration Register **$YFF820**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STOP | 0 | 0 | $\overline{\text{BOOT}}$ | LOCK | EMUL | ASPC | | WAIT | | 0 | 0 | 0 | 0 | 0 | 0 |
| RESET: | | | | | | | | | | | | | | | |
| * | 0 | 0 | USER SPEC | USER SPEC | * | USER SPEC | | USER SPEC | | 0 | 0 | 0 | 0 | 0 | 0 |

*Reset state of STOP = $\overline{\text{DATA14}}$. Reset state of EMUL = ($\overline{\text{DATA10}}$ * $\overline{\text{DATA13}}$).

STOP — Stop Bit
0 = Normal ROM operation
1 = Disable ROM and activate emulator mode if enabled
Reset state of STOP is the complement of DATA14 state during reset. ROM array base address cannot be changed unless STOP is set.

$\overline{\text{BOOT}}$ — Boot ROM Control Bit
0 = CPU16 accesses ROM array addresses after reset
1 = CPU16 cannot access ROM array addresses after reset
Reset state of $\overline{\text{BOOT}}$ is specified by the user. Bootstrap function is overridden if STOP = 1.

LOCK — Lock Registers Bit

0 = Write lock disabled; protected registers and fields can be written

1 = Write lock enabled; protected registers and fields cannot be written

Reset state of LOCK is specified by the user. LOCK protects the ASPC and WAIT fields, as well as the ROMBAL and ROMBAH registers. ASPC, ROMBAL and ROMBAH are also protected by the STOP bit.

EMUL — Emulator Mode Control Bit

0 = Normal ROM operation

1 = MRM enters emulator mode when STOP is set.

Reset state of EMUL is the complement of DATA10 and DATA13 state during reset. When EMUL is set, the MRM responds to accesses by asserting the $\overline{\text{CSM}}$ signal.

ASPC — ROM Array Space Field

Because the CPU16 operates only in supervisory mode, ASPC determines whether accesses are restricted solely to program space, or whether accesses are made to both program and data space. In systems with restricted access levels, ASPC also determines whether accesses are restricted solely to supervisor space. The reset state of ASPC is user specified. The following table shows ASPC encoding.

| ASPC[1:0] | State Specified |
|---|---|
| X0 | Program and data access |
| X1 | Program access only |

WAIT — Wait States Field

WAIT specifies the number of wait states inserted by the MRM during ROM array accesses. It allows the user to optimize bus speed in a particular application by controlling the number of wait states that are inserted before internal $\overline{\text{DSACK}}$ generation. Each wait state has a duration of one system clock cycle. This allows a user to transport code from a slower emulation or development system memory to the ROM array without retiming the system. The reset state of WAIT is user specified. The following table shows WAIT encoding. A no-wait encoding (%00) corresponds to a three clock-cycle bus. The fast termination encoding (%11) corresponds to a two clock-cycle bus. Microcontroller modules typically respond at this rate, but fast termination can also be used to access fast external memory.

| WAIT[1:0] | Cycles per Transfer |
|---|---|
| 00 | 3 |
| 01 | 4 |
| 10 | 5 |
| 11 | 2 |

**ROMBAH** — Array Base Address Register High **$YFF824**

| 15 | 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:

USER SPECIFIED

**ROMBAL** — Array Base Address Register Low **$YFF826**

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

ROMBAH and ROMBAL are used to specify ROM array base address. They can only be written when STOP = 1 and LOCK = 0. This prevents accidental remapping of the array. Since the states of ADDR[23:20] follow the state of ADDR19, addresses in the range $080000 to $F7FFFF cannot be accessed. Because the ROM array must be mapped to a 64-Kbyte boundary, ROMBAL always contains $0000. ROMBAH[7:4], which corresponds to ADDR[23:20], must be initialized to the same value as ROMBAH[3], which corresponds to ADDR[19].

**RSIGHI** — ROM Signature High Register **$YFF828**

| 15 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| NOT USED | | RSP18 | RSP17 | RSP16 |

RESET:

FACTORY SPECIFIED

**RSIGLO** — ROM Signature Low Register **$YFF82A**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RSP15 | RSP14 | RSP13 | RSP12 | RSP11 | RSP10 | RSP9 | RSP8 | RSP7 | RSP6 | RSP5 | RSP4 | RSP3 | RSP2 | RSP1 | RSP0 |

RESET:

FACTORY SPECIFIED

RSIGHI and RSIGLO are used to specify a ROM signature pattern. A special signature identification algorithm can allow the user to verify the content of the ROM array. The signature is specified by the factory and cannot be changed.

**ROMBS0** — ROM Bootstrap Word 0 **$YFF830**

**ROMBS1** — ROM Bootstrap Word 1 **$YFF832**

**ROMBS2** — ROM Bootstrap Word 2 **$YFF834**

**ROMBS3** — ROM Bootstrap Word 3 **$YFF836**

Typically, reset vectors for the system CPU are contained in nonvolatile memory and are only fetched when the CPU comes out of reset. The user can specify that these four words be used as reset vectors, and can specify the content of these locations. The content of these words cannot be changed. In an M68HC16 MCU, ROMBS0 to ROMBS3 correspond to system addresses $000000 to $000006.

# 9 Block-Erasable Flash EEPROM

The 2-Kbyte block-erasable flash EEPROM module (BEFLASH) serves as nonvolatile, fast-access ROM-emulation memory. The module can be used for program code that must either execute at high speed or is frequently executed, such as operating system kernels and standard subroutines, or it can be used for static data that is read frequently. The module can also be configured to provide bootstrap vectors for system reset.

## 9.1 Overview

The BEFLASH module consists of a control-register block that occupies a fixed position in MCU address space, and a 2-Kbyte EEPROM array that can be mapped to any 2-Kbyte boundary in MCU address space. The array can be configured to reside in both program and data space, or in program space alone.

The EEPROM array can be read as either bytes, words, or long-words. The module responds to back-to-back IMB accesses, providing two-bus-cycle (four system clock) access for aligned long words. The module can also be programmed to insert up to three wait states per access, to accommodate migration from slower external development memory without re-timing the system.

Both the array and the individual control bits are programmable and erasable under software control. Program/erase voltage must be supplied through external $V_{FP}$ pins. Programming is by byte or aligned word only. The module supports bulk erase only, and has a minimum program/erase life of 100 cycles. Hardware interlocks protect stored data from corruption if the program/erase voltage to the BEFLASH EEPROM arrày is enabled accidently.

The BEFLASH array is enabled/disabled by a combination of DATA15 and the STOP shadow bit after reset.

**BEFLASH Address Map**

| Address | 15 8 | 7 0 |
|---|---|---|
| $YFF7A0 | BEFLASH MODULE CONFIGURATION REGISTER (BFEMCR) | |
| $YFF7A2 | BEFLASH TEST REGISTER (BFETST) | |
| $YFF7A4 | BEFLASH BASE ADDRESS HIGH REGISTER (BFEBAH) | |
| $YFF7A6 | BEFLASH BASE ADDRESS LOW REGISTER (BFEBAL) | |
| $YFF7A8 | BEFLASH CONTROL REGISTER (BFECTL) | |
| $YFF7AA | RESERVED | |
| $YFF7AC | RESERVED | |
| $YFF7AE | RESERVED | |
| $YFF7B0 | BEFLASH BOOTSTRAP WORD 0 (BFEBS0) | |
| $YFF7B2 | BEFLASH BOOTSTRAP WORD 1 (BFEBS1) | |
| $YFF7B4 | BEFLASH BOOTSTRAP WORD 2 (BFEBS2) | |
| $YFF7B6 | BEFLASH BOOTSTRAP WORD 3 (BFEBS3) | |
| $YFF7B8 | RESERVED | |
| $YFF7BA | RESERVED | |
| $YFF7BC | RESERVED | |
| $YFF7BE | RESERVED | |

Y = M111, where M is the state of the modmap bit in SCIMCR. In an M68HC16 MCU, M must always be set to one.

## 9.2 BEFLASH Control Block

The BEFLASH module control block contains five registers: the BEFLASH module configuration register (BFEMCR), the BEFLASH test register (BFETST), the BEFLASH array base address registers (BFEBAH and BFEBAL), and the BEFLASH control register (BFECTL). Four additional words in the control block can contain bootstrap information when the BEFLASH is used as bootstrap memory.

Each register in the control block has an associated shadow register that is physically located in a spare BEFLASH row. During reset, fields within the registers are loaded with default reset information from the shadow registers. Shadow registers are programmed or erased in the same manner as locations in the BEFLASH array, using the address of the corresponding control registers. When a shadow register is programmed, the data is not written to the corresponding control register. The new data is not copied into the control register until the next reset. The contents of shadow registers are erased whenever the BEFLASH array is erased.

Configuration information is specified and programmed independently of the BEFLASH array. After reset, registers in the control block that contain writable bits can be modified. Writes to these registers do not affect the associated shadow register. Certain registers are writable only when the LOCK bit in BFEMCR is disabled or when the STOP bit in BFEMCR is set. These restrictions are noted in the individual register descriptions.

## 9.3 BEFLASH Array

The base address registers specify the base address of the BEFLASH array. A default reset base address can be programmed into the base address shadow register. The array base address must be on a 2 Kbyte boundary. Because the states of ADDR[23:20] follow the state of ADDR19, addresses in the range $080000 to $F7FFFF cannot be accessed by the CPU16. If the BEFLASH array is mapped to these addresses, the system must be reset before the array can be accessed.

Avoid using a base address value that causes the array to overlap control registers. If a portion of the array overlaps the EEPROM register block, the registers remain accessible, but accesses to that portion of the array are ignored. If the array overlaps the control block of another module, however, those registers may become inaccessible.

## 9.4 BEFLASH Registers

In the following register diagrams, the reset value SB indicates that a bit assumes the value of its associated shadow bit during reset.

**BFEMCR** — BEFLASH Module Configuration Register **$YFF7A0**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STOP | FRZ | 0 | $\overline{\text{BOOT}}$ | LOCK | 0 | ASPC | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RESET: | | | | | | | | | | | | | | | |
| SB | 0 | 0 | SB | SB | 0 | SB | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

This register can be written only when the control block is not write-locked (when LOCK = 0). All active fields and bits take values from the associated shadow register during reset.

STOP — Stop Mode Control

0 = Normal operation

1 = Low-power stop operation

STOP can be set by the processor or by reset if the STOP shadow bit is set. The EEPROM array is inaccessible during low-power stop. The array can be re-enabled by clearing STOP. If STOP is set during programming or erasing, the program/erase voltage is automatically turned off. However, the enable program/erase bit (ENPE) remains set. If STOP is cleared, program/erase voltage is automatically turned back on unless ENPE is cleared.

FRZ — Freeze Mode Control

0 = Disable program/erase voltage while FREEZE is asserted

1 = Allow ENPE bit to turn on the program/erase voltage while FREEZE signal is asserted

$\overline{\text{BOOT}}$ — Boot Control

0 = BEFLASH responds to bootstrap vector addresses after reset

1 = BEFLASH does not respond to bootstrap vector addresses after reset

On reset, the $\overline{\text{BOOT}}$ bit takes on the default value stored in the shadow register. If $\overline{\text{BOOT}}$ = 0 and STOP = 0, the module responds to program space accesses of IMB addresses $000000 to $000006 following reset, and the contents of BFEBS[3:0] are used as bootstrap vectors. After address $000006 is read, the module responds normally to control block or array addresses only.

LOCK — Lock Registers

0 = Write-locking disabled

1 = Write-locked registers protected

If the reset state of the LOCK is zero, it can be set once to protect the registers after initialization. When set, LOCK cannot be cleared until reset occurs.

ASPC — BEFLASH Array Space

Because the CPU16 operates only in supervisory mode, ASPC determines whether accesses are restricted to program space, or whether accesses are made to both program and data space. The field can be written to only if LOCK = 0 and STOP = 1. During reset, ASPC takes on the default value programmed into the associated shadow register.

| ASPC[1:0] | Type of Access |
|---|---|
| X0 | Program and data |
| X1 | Program only |

ASPC assigns the BEFLASH array to supervisor or user space, and to program or data space.

**BFETST** — BEFLASH Test Register **$YFF7A2**

This register is used for factory test purposes only.

**BFEBAH** — BEFLASH Base Address High Register **$YFF7A4**

| 15 |  |  |  |  |  |  | 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: |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | SHADOW BIT DEFAULT VALUE |  |  |  |  |  |  |  |

**BFEBAL** — BEFLASH Base Address Low Register **$YFF7A6**

| 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RESET: |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

BFEBAH and BFEBAL contain the 13 high-order bits of the BEFLASH array base address. During reset, BFEBAH takes on the default value programmed into the associated shadow register. After reset, if LOCK = 0 and STOP = 1, software can write to BFEBAH and BFEBAL to relocate the BEFLASH array. Because the states of ADDR[23:20] follow the state of ADDR19, addresses in the range $080000 to $F7FFFF cannot be accessed by the CPU16. If the BEFLASH array is mapped to these addresses, the system must be reset before the array can be accessed.

**BFECTL** — BEFLASH Control Register **$YFF7A8**

| 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 | VFPE | ERAS | LAT | ENPE |
| RESET: |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

BFECTL contains the bits needed to control programming and erasing the BEFLASH.

VFPE — Verify Program/Erase

0 = Normal read cycles
1 = Invoke program-verify circuit

This bit invokes a special program-verify circuit. During programming sequences (ERAS = 0), VFPE is used in conjunction with the LAT bit to determine when programming of a location is complete. If VFPE and LAT are both set, a bit-wise exclusive-OR of the latched data with the data in the location being programmed occurs when any valid BEFLASH location is read. If the location is completely programmed, a value of zero is read. Any other value indicates that the location is not fully programmed. When VFPE is cleared, normal reads of valid BEFLASH locations occur.

ERAS — Erase Control

0 = BEFLASH configured for programming
1 = BEFLASH configured for erasure

Asserting ERAS causes all locations in the array and all BEFLASH shadow bits in the control block to be configured for erasure at the same time.

When the LAT bit is set, ERAS also determines whether a read returns the value of the addressed location (ERAS = 1) or the location being programmed (ERAS = 0).

The value of ERAS cannot be changed if the program/erase voltage is turned on (ENPE = 1).

LAT — Latch Control

0 = Programming latches disabled
1 = Programming latches enabled

When LAT is cleared, the BEFLASH address and data buses are connected to the IMB address and data buses and the BEFLASH is configured for normal reads. When LAT is set, the BEFLASH address and data buses are connected to parallel internal latches and the BEFLASH array is configured for programming or erasing.

Once LAT is set, the next write to a valid BEFLASH address causes the programming circuitry to latch both address and data. Unless control register shadow bits are to be programmed, the write must be to an array address.

The value of LAT cannot be changed when program/erase voltage is turned on (ENPE = 1).

ENPE — Enable Program/Erase

0 = Disable program/erase voltage
1 = Apply program/erase voltage

ENPE can be set only after LAT has been set and a write to the data and address latches has occurred. ENPE remains cleared if these conditions are not met. While ENPE is set, the LAT, VFPE, and ERAS bits cannot be changed, and attempts to read a BEFLASH array location in BEFLASH are ignored.

**BFEBS[3:0]** — BEFLASH Bootstrap Words **$YFF7B0 – $YFF7B6**

| 15 | 0 |
|---|---|
| BOOTSTRAP VECTOR | |
| RESET: | |
| PROGRAMMED VALUE | |

These words can be used as system bootstrap vectors. When the $\overline{\text{BOOT}}$ bit in BFEMCR = 0 during reset, the BEFLASH responds to program space accesses of IMB addresses $000000 to $000006 after reset. When $\overline{\text{BOOT}}$ = 1, the BEFLASH responds only to normal array and register accesses. BFEBS[3:0] can be read at any time, but the values in the words can only be changed by programming the appropriate location.

## 9.5 BEFLASH Operation

The following paragraphs describe BEFLASH reset, using the module for system bootstrap, normal operation, and array programming and erasing.

### 9.5.1 Reset Operation

Reset initializes all BEFLASH control registers. Some bits have fixed default values, and some take values that are programmed into the associated BEFLASH shadow registers.

When the state of the STOP shadow bit is zero, the STOP bit in BFEMCR is cleared during reset, and the module responds to accesses in the range specified by BFEBAH and BFEBAL. When the BOOT bit is cleared, the module also responds to bootstrap vector accesses.

When the state of the STOP shadow bit is one, the STOP bit in BFEMCR is set during reset and the BEFLASH array is disabled. The module does not respond to array or bootstrap vector accesses until the STOP bit is cleared. This allows an external device to respond to accesses to the BEFLASH array address space or to bootstrap accesses. The erased state of the shadow bits is one. An erased module comes out of reset in STOP mode.

### 9.5.2 Bootstrap Operation

The CPU16 begins bootstrap operation by fetching initial values for its internal registers from IMB addresses $000000 through $000006 in program space. These are the addresses of the bootstrap vectors in the exception vector table. If the $\overline{\text{BOOT}}$ and STOP bits in BFEMCR are cleared during reset, the BEFLASH module is configured to respond to bootstrap vector accesses. Vector assignments are as follows:

| EEPROM Bootstrap Word | IMB Vector Address | MCU Reset Vector Content |
|---|---|---|
| BFEBS0 | $000000 | Initial ZK, SK, and PK |
| BFEBS1 | $000002 | Initial PC |
| BFEBS2 | $000004 | Initial SP |
| BFEBS3 | $000006 | Initial IZ |

As soon as address $000006 has been read, BEFLASH operation returns to normal, and the module no longer responds to bootstrap vector accesses.

### 9.5.3 Normal Operation

The BEFLASH module performs byte or aligned-word accesses in one bus cycle. Long-word reads or writes require an additional bus cycle. The WAIT field in BFEMCR can be used to insert wait states.

The module checks function codes to verify address-space access type. Array accesses are defined by the state of ASPC in BFEMCR. When the BEFLASH is configured for normal operation, the array responds to read accesses only; write operations are ignored.

Accesses to an address in the 64-Kbyte block defined by the base address registers that does not fall within the array (the upper 16 Kbytes of the block) are driven externally, allowing an external device to fill the entire address space defined by the base address.

### 9.5.4 Program/Erase Operation

An unprogrammed EEPROM bit has a logic state of one. A bit must be programmed to change its state from one to zero. Erasing a bit returns it to the state of one. Programming or erasing the BEFLASH array requires a series of control register writes and a write to a set of programming latches. The same procedure is used to program array locations and control registers that contain EEPROM bits. Programming is restricted to a single byte or aligned word at a time. The entire BEFLASH array and the shadow register bits are erased at the same time.

**NOTE**

In order to program the array, programming voltage must be applied to the $V_{PP}$ pin. $V_{PP} \geq (V_{DD} - 0.3\text{ V})$ must be applied at all times or damage to the BEFLASH module can occur.

### 9.5.4.1 Intelligent Programming and Erasing

Intelligent programming and erasing procedures verify the BEFLASH array as it is being altered. This ensures accurate results and provides the longest possible life expectancy for the module. The user must stop the programming or erase sequence at periods of $t_{ppulse}$ or $t_{epulse}$ to determine whether a sequence was executed successfully. The $t_{ppulse}$ or $t_{epulse}$ values must be recalculated after each pulse for optimum performance. After a location reaches the proper value, the programming/erasure cycle must continue for a short period ($t_{pmargin}$ or $t_{emargin}$) to ensure that the value is made permanent. Use the following procedures for programming and erasing the BEFLASH.

### 9.5.4.2 Programming Sequence

1. Turn on $V_{fp}$ (apply $V_{fp}$ to $V_{FPE48K}$ pin).
2. Clear ERAS and set LAT and VFPE bits in BFECTL to set program mode, enable programming address and data latches, and invoke special verification read circuitry. Set initial value of $t_{ppulse}$ to $t_{pmin}$.
3. Write new data to the desired address. This causes the address and data of the location to be programmed to be latched in the programming latches.
4. Set ENPE to apply programming voltage.
5. Delay long enough for one programming pulse to occur ($t_{ppulse}$).
6. Clear ENPE to remove programming voltage.
7. Delay while high voltage is turning off ($t_{vprog}$).
8. Read the location just programmed. If the value read is all zeros, proceed to step 9. If not, calculate a new value for $t_{ppulse}$ and repeat steps 4 through 7 until either the location is verified or the total programming time ($t_{progmax}$) has been exceeded. If $t_{progmax}$ has been exceeded, the location may be bad and should not be used.
9. If the location is programmed, calculate $t_{pmargin}$ and repeat steps 4 through 7. If the location does not remain programmed, the location is bad.
10. Clear VFPE and LAT.
11. If there are more locations to program, repeat steps 2 through 10.
12. Turn off $V_{fp}$ (reduce voltage on $V_{FPE48K}$ pin to $V_{DD}$).
13. Read the entire array to verify that all locations are correct. If any locations are incorrect, the array is bad.

#### 9.5.4.3 Erasure Sequence

1. Turn on $V_{fp}$ (apply $V_{fp}$ to $V_{FPE48K}$ pin).
2. Set LAT, VFPE, and ERAS bits to configure the BEFLASH module for erasure. Set initial value of $t_{epulse}$ to $t_{emin}$.
3. Write to any valid address in the control block or array. This allows the erase voltage to be turned on. The data written and the address written to are of no consequence.
4. Set ENPE to apply programming voltage.
5. Delay long enough for one erase pulse to occur ($t_{epulse}$).
6. Clear ENPE to remove programming voltage.
7. Delay while high voltage is turning off ($t_{vprog}$).
8. Read the entire array and control block to ensure that the entire module is erased.
9. If all of the locations are not erased, calculate a new value for $t_{epulse}$ and repeat steps 4 through 8 until either the remaining locations are erased or the maximum erase time ($t_{erase}$) has been exceeded. If $t_{erase}$ has been exceeded, the location may be bad and should not be used.
10. If all locations are erased, calculate $t_{emargin}$ and repeat steps 4 through 8. If all locations do not remain erased, the BEFLASH module may be bad.
11. Clear LAT, ERAS, and VFPE to allow normal access to the BEFLASH.
12. Turn off $V_{fp}$ (reduce voltage on $V_{FPE48K}$ pin to $V_{DD}$).

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and Ⓜ are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

**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.

MOTOROLA

1ATX31348-0 PRINTED IN USA 5/93 IMPERIAL LITHO 91760 18,000 MCU YGACAA

MC68HC16X1TS/D