MC68HC916Y1

Technical Summary
16-Bit Modular Microcontroller

1 Introduction

The MC68HC916Y1 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 MC68HC916Y1 incorporates a true 16-bit CPU (CPU16), a single-chip integration module (SCIM), an 8/10-bit analog-to-digital converter (ADC), a multichannel communication interface (MCCI), a general-purpose timer (GPT), a time processing unit (TPU), a 2-Kbyte standby RAM module with TPU ROM emulation capability (TPURAM), a 2-Kbyte standby RAM module with no TPU ROM emulation capability (STBRAM), and a 48-Kbyte flash EEPROM module (FLASH). The MC68HC916Y1 has special features that facilitate emulation of the MC68HC16Y1.

M68HC16 devices can either synthesize an internal clock signal from an external reference, or use an external clock input directly. Operation with a 4.194-MHz reference frequency is standard. System hardware and software allow changes in clock rate during operation — register and memory contents are not affected by clock rate changes.

High-density complementary metal-oxide semiconductor (HCMOS) architecture makes the basic power consumption of M68HC16 devices 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
Plastic Surface Mount	16.78	–40° to +125°C	M68HC916Y1CFC

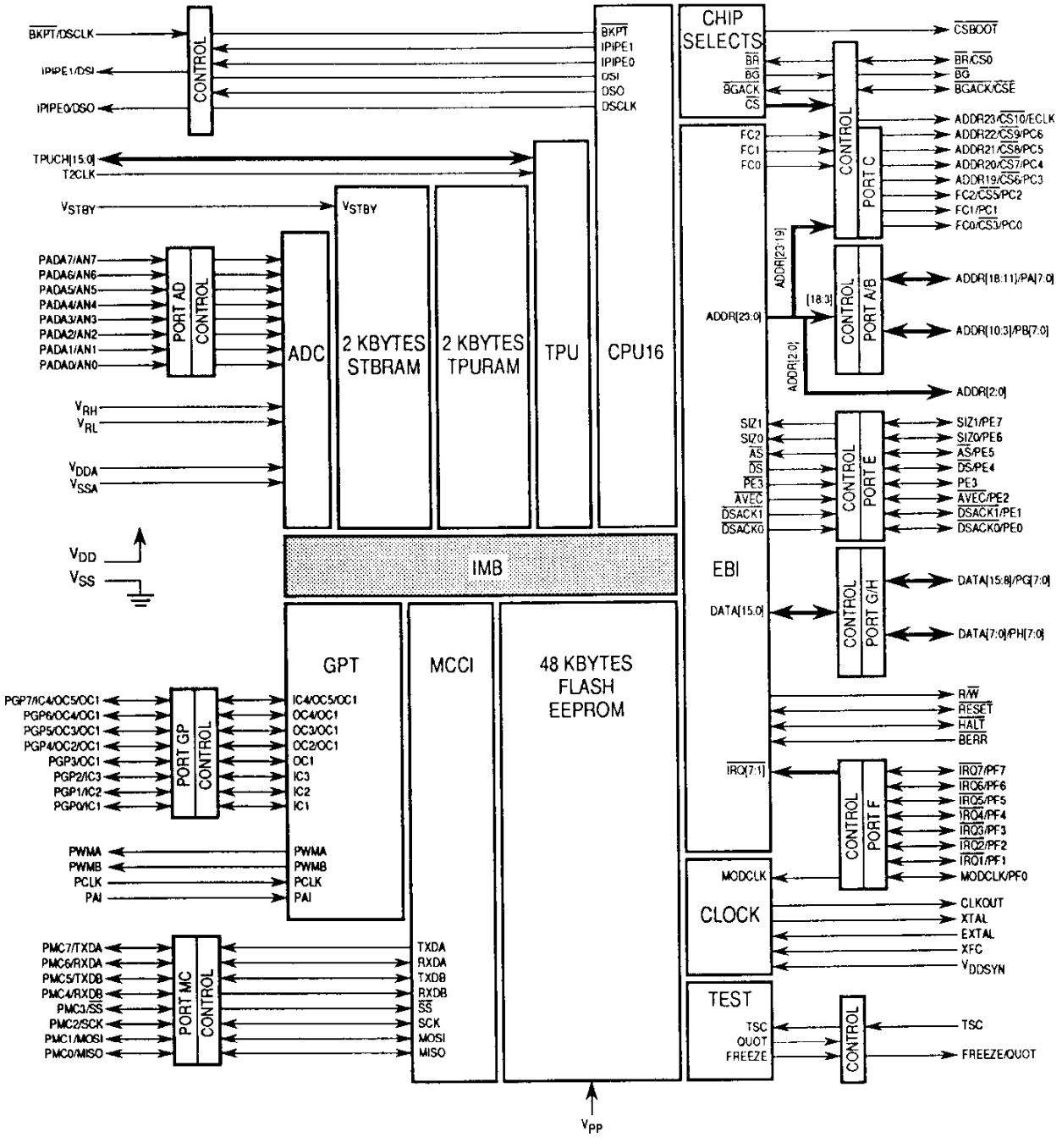
This document contains information about a new product. Specifications and information are subject to change without notice.

Table of Contents

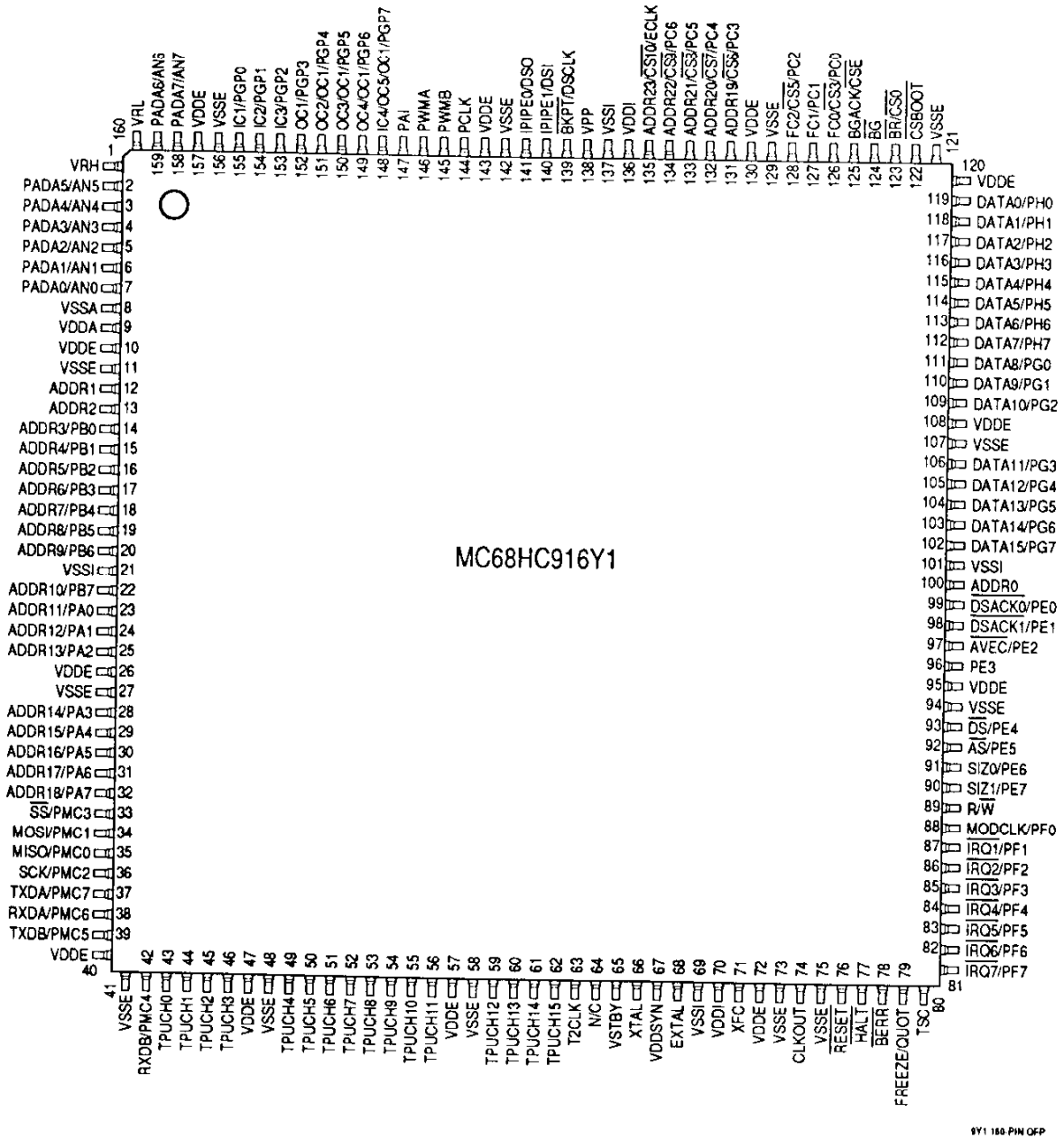
Table of Contents (Continued)

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
  - Emulation-Support Chip Select Output
  - 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
- Time Processor Unit (TPU)
  - Dedicated Microengine Operating Independently of CPU16
  - 16 Independently Programmable Channels and Pins
  - Two Timer Count Registers with Programmable Prescalers
  - Selectable Channel Priority Levels
- 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)
  - Eight Channels, Eight Result Registers
  - Eight Automated Modes
  - Three Result-Alignment Modes
- Multichannel Communication Interface (MCCI)
  - Dual Serial Communication Interface (SCI)
  - Serial Peripheral Interface (SPI)
- TPU Emulation RAM Module (TPURAM)
  - 2 Kbyte Static RAM Array, Mappable to any 2 Kbyte Boundary
  - TPU Microcode Emulation
- Standby RAM Module (STBRAM)
  - 2 Kbyte Static RAM Array, Mappable to any 2 Kbyte Boundary
  - External Standby Voltage Supply Input and Power-Loss Flag
- Flash EEPROM Module (FLASH)
  - 48 Kbyte, Bulk-Erasable 16-Bit Array
  - Boot ROM Capability



MC68HC916Y1 Block Diagram



160-Pin Quad Flat Pack Pin Assignments

1.1 Pin Description

The following table contains 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 **Driver Types** table for a description of output drivers. An entry in the discrete I/O column of the **Pin Characteristics** table indicates that a pin has an alternate I/O function. The port designation is given when it applies. Refer to the **MC68HC916Y1 Block Diagram** for port organization.

Pin Characteristics

Pin Mnemonic	Output Driver	Input Synchronized	Input Hysteresis	Discrete I/O	Port Designation
ADDR23/$\overline{CS10}$/ECLK	A	Y	N	—	—
ADDR[22:19]/$\overline{CS[9:6]}$	A	Y	N	O	C[6:3]
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[7:0][1]	—	Y	Y	I	ADA[7:0]
$\overline{AS}$	B	Y	Y	I/O	E5
$\overline{AVEC}$	B	Y	N	I/O	E2
$\overline{BERR}$	B	Y	N	—	—
$\overline{BG}$	B	—	—	—	—
$\overline{BGACK}$/$\overline{CSE}$	B	Y	N	—	—
$\overline{BKPT}$/DSCLK	—	Y	Y	—	—
$\overline{BR}$/$\overline{CS0}$	B	Y	N	—	—
CLKOUT	A	—	—	—	—
$\overline{CSBOOT}$	B	—	—	—	—
DATA[15:8][1]	Aw	Y	Y	I/O	G[7:0]
DATA[7:0][1]	Aw	Y	Y	I/O	H[7:0]
$\overline{DS}$	B	Y	Y	I/O	E4
$\overline{DSACK1}$	B	Y	N	I/O	E1
$\overline{DSACK0}$	B	Y	N	I/O	E0
DSI/IPIPE1	A	Y	Y	—	—
DSO/IPIPE0	A	—	—	—	—
EXTAL[2]	—	—	—	—	—
FC2/$\overline{CS5}$	A	Y	N	O	C0
FC1	A	Y	N	O	C1
FC0/$\overline{CS3}$	A	Y	N	O	C2
FREEZE/QUOT	A	—	—	—	—
$\overline{HALT}$	Bo	Y	N	—	—
IC4/OC5	A	Y	Y	I/O	GP7
IC[3:1]	A	Y	Y	I/O	GP[2:0]
$\overline{IRQ}$[7:1]	B	Y	Y	I/O	F[7:1]

Pin Characteristics (Continued)

Pin Mnemonic	Output Driver	Input Synchronized	Input Hysteresis	Discrete I/O	Port Designation
MISO	Bo	Y	Y	I/O	MC0
MODCLK[1]	B	Y	Y	I/O	F0
MOSI	Bo	Y	Y	I/O	MC1
OC[4:1]	A	Y	Y	I/O	GP[6:3]
PAI[3]	—	Y	Y	I	—
PCLK[3]	—	Y	Y	I	—
PE3	B	Y	Y	I/O	E3
PWMA, PWMB[4]	A	Y	Y	O	—
R/$\overline{W}$	A	Y	N	—	—
$\overline{RESET}$	Bo	Y	Y	—	—
RXDA	Bo	Y	Y	I/O	MC6
RXDB	Bo	Y	Y	I/O	MC4
SCK	Bo	Y*	Y	I/O	MC2
SIZ[1:0]	B	Y	N	I/O	E[7:6]
$\overline{SS}$	Bo	Y	Y	I/O	MC3
TSC	—	Y	Y	—	—
TPUCH[15:0]	A	Y	Y	—	—
T2CLK	—	Y	Y	—	—
TXDA	Bo	Y	Y	I/O	MC7
TXDB	Bo	Y	Y	I/O	MC5
V_{RH}[5]	—	—	—	—	—
V_{RL}[5]	—	—	—	—	—
XFC[2]	—	—	—	—	—
XTAL[2]	—	—	—	—	—

NOTES

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

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.

Power Connections

V_{DDA}/V_{SSA}	A/D Converter Power
V_{DDSYN}	Clock Synthesizer Power
V_{SSE}/V_{DDE}	External Peripheral Power (Source and Drain)
V_{STBY}	Standby RAM Power/Clock Synthesizer Power
V_{PP}	EEPROM Array Program/Erase Power

1.2 Signal Description

The following tables indicate MCU signal type and function.

Signal Characteristics

Signal Name	MCU Module	Signal Type	Active State
ADA[7:0]	ADC	Input	—
ADDR[23:0]	SCIM	Bus	—
$\overline{AS}$	SCIM	Output	0
$\overline{AVEC}$	SCIM	Input	0
$\overline{BERR}$	SCIM	Input	0
$\overline{BG}$	SCIM	Output	0
$\overline{BGACK}$	SCIM	Input	0
$\overline{BKPT}$	CPU16	Input	0
$\overline{BR}$	SCIM	Input	0
CLKOUT	SCIM	Output	—
$\overline{CS[10:5]}$, $\overline{CS3}$, $\overline{CS0}$	SCIM	Output	0
$\overline{CSBOOT}$	SCIM	Output	0
$\overline{CSE}$	SCIM	Output	0
DATA[15:0]	SCIM	Bus	—
$\overline{DS}$	SCIM	Output	0
$\overline{DSACK[1:0]}$	SCIM	Input	0
DSCLK	CPU16	Input	Serial Clock
DSI	CPU16	Input	(Serial Data)
DSO	CPU16	Output	(Serial Data)
ECLK	CPU16	Output	—
EXTAL	SCIM	Input	—
FC[2:0]	SCIM	Output	—
FREEZE	SCIM	Output	1
$\overline{HALT}$	SCIM	Input/Output	0
IC[4:1]	GPT	Input	—
IPIPE0	CPU16	Output	—
IPIPE1	CPU16	Output	—
$\overline{IRQ[7:1]}$	SCIM	Input	0
MISO	MCCI	Input/Output	—
MODCLK	SCIM	Input	—
MOSI	MCCI	Input/Output	—
OC[5:1]	GPT	Output	—
PAI	GPT	Input	—

Signal Characteristics (Continued)

Signal Name	MCU Module	Signal Type	Active State
PCLK	GPT	Input	—
PWMA, PWMB	GPT	Output	—
QUOT	SCIM	Output	—
R/$\overline{W}$	SCIM	Output	—
$\overline{RESET}$	SCIM	Input/Output	0
RXDA	MCCI	Input	—
RXDB	MCCI	Input	—
SCK	MCCI	Input/Output	—
SIZ0/SIZ1	SCIM	Output	—
$\overline{SS}$	MCCI	Input	0
TSC	SCIM	Input	1
TPUCH[15:0]	TPU	Input/Output	—
T2CLK	TPU	Input	—
TXDA	MCCI	Output	—
TXDB	MCCI	Output	—
XFC	SCIM	Input	—
XTAL	SCIM	Output	—

Signal Function

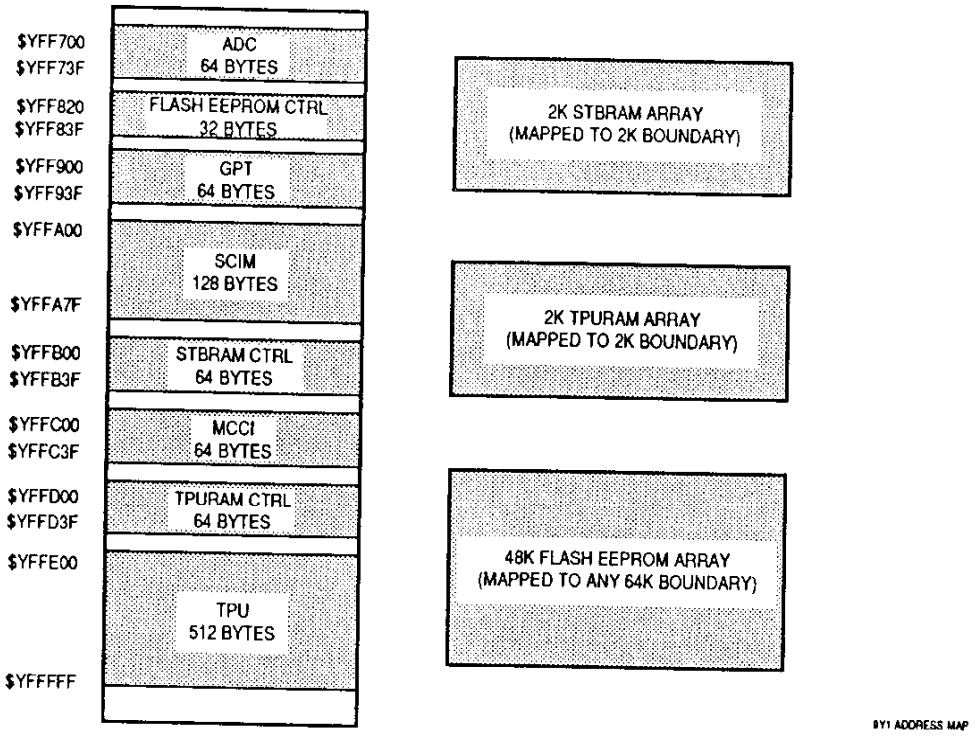
Signal Name	Mnemonic	Function
ADC Analog Input	ADA[7:0]	Inputs to ADC MUX
Address Bus	ADDR[19:0]	20-bit address bus used by CPU16
Address Bus	ADDR[23:20]	4 MSB on IMB, outputs follow ADDR19
Address Strobe	$\overline{AS}$	Indicates that a valid address is on the address bus
Autovector	$\overline{AVEC}$	Requests an automatic vector during interrupt acknowledge
Bus Grant	$\overline{BG}$	Indicates that the MCU has relinquished the bus
Bus Grant Acknowledge	$\overline{BGACK}$	Indicates that an external device has assumed bus mastership
Bus Error	$\overline{BERR}$	Indicates that a bus error has occurred
Breakpoint	$\overline{BKPT}$	Signals a hardware breakpoint to the CPU
Bus Request	$\overline{BR}$	Indicates that an external device requires bus mastership
System Clockout	CLKOUT	System clock output
Emulation Mode Chip Select	$\overline{CSE}$	$\overline{CSE}$ selects external emulation devices at internally-mapped addresses and is used to emulate I/O ports.
General-Purpose Chip Selects	$\overline{CS[10:5]}$, $\overline{CS3}$, $\overline{CS0}$	Select external devices at programmed addresses
Boot Chip Select	$\overline{CSBOOT}$	Chip select for external boot startup ROM
Data Bus	DATA[15:0]	16-bit data bus
Data Strobe	$\overline{DS}$	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	$\overline{DSACK[1:0]}$	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
Halt	$\overline{HALT}$	Suspend external bus activity
Instruction Pipeline	IPIPE0 IPIPE1	Indicate instruction pipeline activity
Interrupt Request	$\overline{IRQ[7:1]}$	Request interrupt service from CPU16
Master In Slave Out	MISO	Serial input to SPI in master mode; 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; serial input to SPI in slave mode

Signal Function (Continued)

Signal Name	Mnemonic	Function
Port A	PA[0:7]	SCIM digital I/O port signals
Port ADA	PADA[7:0]	ADC digital input port signals
Port B	PB[0:7]	SCIM digital I/O port signals
Port C	PC[6:0]	SCIM digital output port signals
Port E	PE[7:0]	SCIM digital I/O port signals
Port F	PF[7:0]	SCIM digital I/O port signals
Port G	PG[7:0]	SCIM digital I/O port signals
Port GP	PGP[7:0]	GPT digital I/O port signals
Port H	PH[7:0]	SCIM digital I/O port signals
Port MC	PMC[7:0]	MCCI digital I/O port signals
Quotient Out	QUOT	Provides the quotient bit of the polynomial divider
Read/Write	R/$\overline{W}$	Indicates the direction of data transfer on the bus
Reset	$\overline{RESET}$	System reset
SCI A Receive Data	RXDA	Serial input to SCI A
SCI B Receive Data	RXDB	Serial input to SCI B
SPI Serial Clock	SCK	Clock output from SPI in master mode; clock input to SPI in slave mode
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
TPU Channels	TPU CH[15:0]	Independently programmable timer channels
TPU Clock	T2CLK	External TPU clock input
SCI A Transmit Data	TXDA	Serial output from SCI A
SCI B Transmit Data	TXDB	Serial output from SCI B
External Filter Capacitor	XFC	Connection for external phase-locked loop filter capacitor

1.3 Address Map

The following figure illustrates the internal address map of the MCU. Although there are 24 intermodule bus (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.



System Address Map

In the address map, Y = M111, where M represents the state of the MODMAP (MM) bit in the single-chip integration module configuration register (SCIMCR). In M68HC16 devices, Y must equal $F. If MM is cleared, IMB modules are inaccessible until a reset occurs. MM can be written only once after reset.

1.4 Intermodule Bus

The intermodule bus (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 driven to the same logic state as ADDR19. ADDR[23:20] are brought out to pins for test purposes.

1.5 Using the MC68HC916Y1 to Emulate the MC68HC16Y1

The MC68HC916Y1 is designed to provide MC68HC16Y1 emulation capabilities. All common functions in the two devices operate identically. However, there are differences that must be taken into account.

1.5.1 Flash EEPROM and Masked ROM

The control registers of the flash EEPROM module in the MC68HC916Y1 occupy the same addresses as the control registers of the MC68HC16Y1 masked ROM module, and the FLASH provides boot ROM capability. However, care must be taken to correctly program the FLASH shadow registers to the same values as the emulated masked ROM registers.

In the MC68HC16Y1, external ROM emulation is enabled by holding DATA10 and DATA13 low during reset (DATA14 must be held high during reset to enable the ROM module). While ROM emulation mode is enabled, memory chip select signal $\overline{\text{CSM}}$ is asserted whenever a valid access to an address assigned to the masked ROM array is made. Because the MC68HC916Y1 has no ROM, the $\overline{\text{CSM}}$ function is not used — the $\overline{\text{CSM}}$ pin is driven high whenever the function is selected.

The user must supply flash EEPROM programming voltage to the MC68HC916Y1. The V_{PP} connection is provided on a pin that is not used on the MC68HC16Y1. $V_{PP} \geq (V_{DD} - 0.3\ V)$ must be applied at all times or damage to the FLASH module can occur.

1.5.2 STBRAM and TPURAM

The MC68HC916Y1 has one more 2-Kbyte RAM module than the MC68HC16Y1. The two 2-Kbyte RAM modules in the MC68HC916Y1 are structurally similar, but functionally different. The TPURAM module has no external standby voltage (V_{STBY}) connection or power-loss flag (PDS), but supports the use of custom TPU microcode. The STBRAM module, on the other hand, has a V_{STBY} connection and provides a power-loss flag and automatic switching to standby power when V_{DD} drops below a specified level, but does not support TPU microcode emulation.

MC68HC916Y1 TPURAM control registers are located at addresses $YFFD00–YFFD3F, while STBRAM module control registers are located from $YFFB00–YFFB3F. MC68HC916Y1 STBRAM control registers occupy the same addresses as MC68HC16Y1 TPURAM control registers. MC68HC916Y1 TPURAM control register addresses are in different locations from the MC68HC16Y1 TPURAM control registers.

The TPURAM array can be mapped to form a contiguous extension of the STBRAM array. While it is possible to map STBRAM over TPURAM while TPURAM is used for microcode emulation, this is not recommended, as this effectively makes a wired-AND connection between the module data bus lines, and can affect accesses to STBRAM.

2 CPU16

The CPU16 is a true 16-bit high-speed device. It was designed to give M68HC11 users a way of gaining 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 simplified by the use of the available 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 for implementing control-oriented digital signal processing functions with minimum interfacing. A multiply and accumulate (MAC) 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.

The CPU16 instruction set supports high-level languages, the use of which is increasing as controller applications become more complex and control programs become larger. These high-level languages aid the rapid and reduced-error development of readily portable software.

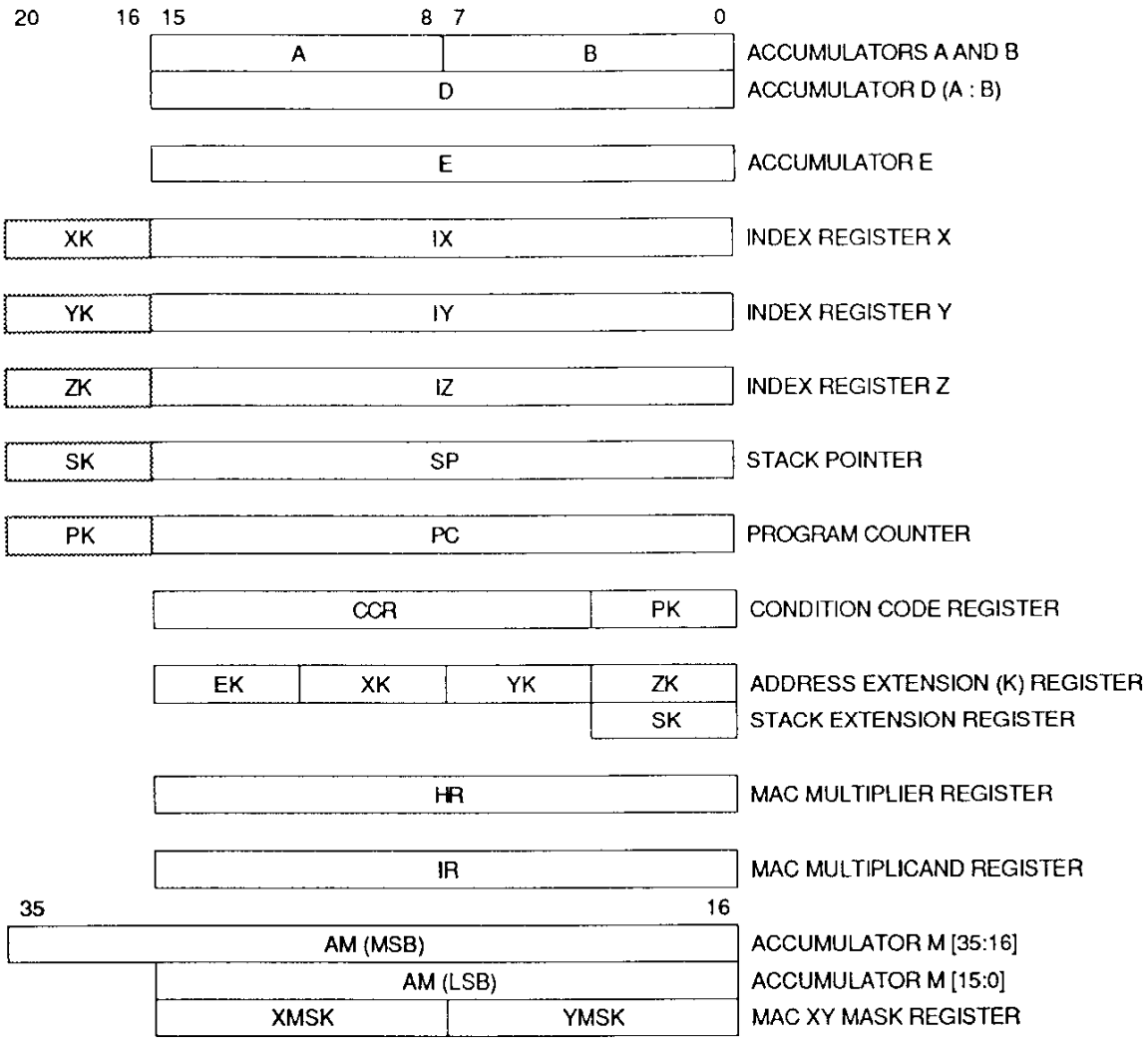
2.2 M68HC11 Compatibility

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.

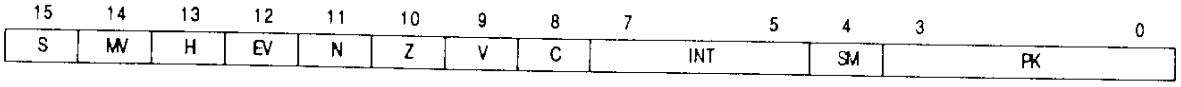
The CPU16 also has several new or enhanced addressing modes. M68HC11 direct mode addressing has been replaced by a special form of indexed addressing that uses the new IZ register and a reset vector to provide greater flexibility.

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
Accumulator M — 36-bit MAC result 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
XMSK, YMSK — Determine which bits change when an offset is added

2.4 Condition Code Register



The condition code register (CCR) can be considered as two functional blocks. The most significant bit (MSB), which corresponds to the CCR in the M68HC11, contains the low-power stop control bit and processor status flags. The least significant bit (LSB) contains the interrupt priority field, the digital signal processing (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.

2.5 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 can 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.

2.6 Addressing Modes

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

All addressing 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 usually does not change when an access crosses a bank boundary.

In 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, which 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 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 therefore are 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 the 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 index 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 for the loss of direct mode, the ZK field and index register Z have been assigned reset initialization vectors. By resetting the ZK field to a chosen page and using 8-bit unsigned index mode with IZ, a programmer can access useful data structures anywhere in the address map.

2.7 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 has 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 may be executed differently. Most M68HC11 code runs on the CPU16 following reassembly. The user must take into account changed instruction times, a different arbitration scheme, and a 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.

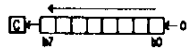
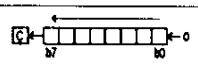
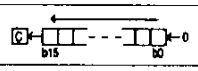
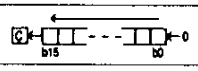
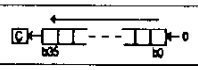
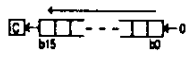
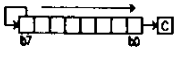
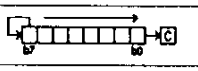
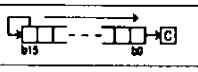
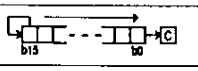
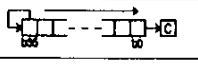
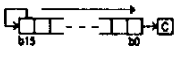
Instruction Set Summary

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

Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
ADDB	Add to B	(B) + (M) ⇒ B	IND8, X	C1	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 Mode	Instruction Opcode	Instruction Operand	Instruction Cycles	Condition Codes S	MV	H	EV	N	Z	V	C
ANDD	AND D	(D) • (M : M + 1) ⇒ D	IND8, X	86	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	96	ff	6								
			IND8, Z	A6	ff	6								
			E, X	2786	—	6								
			E, Y	2796	—	6								
			E, Z	27A6	—	6								
			IMM16	37B6	jj kk	4								
			IND16, X	37C6	gggg	6								
			IND16, Y	37D6	gggg	6								
			IND16, Z	37E6	gggg	6								
			EXT	37F6	hh ll	6								
ANDE	AND E	(E) • (M : M + 1) ⇒ E	IMM16	3736	jj kk	4	—	—	—	—	Δ	Δ	0	—
			IND16, X	3746	gggg	6								
			IND16, Y	3756	gggg	6								
			IND16, Z	3766	gggg	6								
			EXT	3776	hh ll	6								
ANDP[1]	AND CCR	(CCR) • IMM16⇒ CCR	IMM16	373A	jj kk	4	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
ASL	Arithmetic Shift Left		IND8, X	04	ff	8	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	14	ff	8								
			IND8, Z	24	ff	8								
			IND16, X	1704	gggg	8								
			IND16, Y	1714	gggg	8								
			IND16, Z	1724	gggg	8								
			EXT	1734	hh ll	8								
ASLA	Arithmetic Shift Left A		INH	3704	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASLB	Arithmetic Shift Left B		INH	3714	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASLD	Arithmetic Shift Left D		INH	27F4	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASLE	Arithmetic Shift Left E		INH	2774	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASLM	Arithmetic Shift Left AM		INH	27B6	—	4	—	Δ	—	Δ	Δ	—	—	Δ
ASLW	Arithmetic Shift Left Word		IND16, X	2704	gggg	8	—	—	—	—	Δ	Δ	Δ	Δ
			IND16, Y	2714	gggg	8								
			IND16, Z	2724	gggg	8								
			EXT	2734	hh ll	8								
ASR	Arithmetic Shift Right		IND8, X	0D	ff	8	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	1D	ff	8								
			IND8, Z	2D	ff	8								
			IND16, X	170D	gggg	8								
			IND16, Y	171D	gggg	8								
			IND16, Z	172D	gggg	8								
			EXT	173D	hh ll	8								
ASRA	Arithmetic Shift Right A		INH	370D	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASRB	Arithmetic Shift Right B		INH	371D	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASRD	Arithmetic Shift Right D		INH	27FD	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASRE	Arithmetic Shift Right E		INH	277D	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASRM	Arithmetic Shift Right AM		INH	27BA	—	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 Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
BCLR	Clear Bit(s)	(M) • ($\overline{\text{Mask}}$) ⇒ M	IND16, X	08	mm gggg	8	—	—	—	—	Δ	Δ	0	—
			IND16, Y	18	mm gggg	8								
			IND16, Z	28	mm gggg	8								
			EXT	38	mm hh ll	8								
			IND8, X	1708	mm ff	8								
			IND8, Y	1718	mm ff	8								
			IND8, Z	1728	mm ff	8								
BCLRW	Clear Bit(s) Word	(M : M + 1) • ($\overline{\text{Mask}}$) ⇒ M : M + 1	IND16, X	2708	gggg mmmm	10	—	—	—	—	Δ	Δ	0	—
			IND16, Y	2718	gggg mmmm	10								
			IND16, Z	2728	gggg mmmm	10								
			EXT	2738	hh ll mmmm	10								
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	49	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	59	ff	6								
			IND8, Z	69	ff	6								
			IMM8	79	ii	2								
			IND16, X	1749	gggg	6								
			IND16, Y	1759	gggg	6								
			IND16, Z	1769	gggg	6								
			EXT	1779	hh ll	6								
			E, X	2749	—	6								
			E, Y	2759	—	6								
			E, Z	2769	—	6								
BITB	Bit Test B	(B) • (M)	IND8, X	C9	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	D9	ff	6								
			IND8, Z	E9	ff	6								
			IMM8	F9	ii	2								
			IND16, X	17C9	gggg	6								
			IND16, Y	17D9	gggg	6								
			IND16, Z	17E9	gggg	6								
			EXT	17F9	hh ll	6								
			E, X	27C9	—	6								
			E, Y	27D9	—	6								
			E, Z	27E9	—	6								
BLE[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	CB	mm ff rr	10, 12	—	—	—	—	—	—	—	—
			IND8, Y	DB	mm ff rr	10, 12								
			IND8, Z	EB	mm ff rr	10, 12								
			IND16, X	0A	mm gggg rr	10, 14								
			IND16, Y	1A	mm gggg rr	10, 14								
			IND16, Z	2A	mm gggg rr	10, 14								
			EXT	3A	mm hh ll rr	10, 14								
BRN	Branch Never	If 1 = 0, branch	REL8	B1	rr	2	—	—	—	—	—	—	—	—

Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction Opcode	Instruction Operand	Instruction Cycles	Condition Codes S	MV	H	EV	N	Z	V	C
BRSET[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 mm	10, 14								
			IND16, Y	1B	mm gggg mm	10, 14								
			IND16, Z	2B	mm gggg mm	10, 14								
			EXT	3B	mm hh ll mm	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 Push (PC) (SK : SP) − 2 ⇒ SK : SP Push (CCR) (SK : SP) − 2 ⇒ SK : SP (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 Mode	Instruction			Condition Codes							
				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 Mode	Instruction		Cycles	Condition Codes							
				Opcode	Operand		S	MV	H	EV	N	Z	V	C
CPZ	Compare IZ to Memory	(IZ) – (M : M + 1)	IND8, X	4E	ff	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND8, Y	5E	ff	6								
			IND8, Z	6E	ff	6								
			IND16, X	174E	gggg	6								
			IND16, Y	175E	gggg	6								
			IND16, Z	176E	gggg	6								
			EXT	177E	hh ll	6								
			IMM16	377E	jj kk	4								
DAA	Decimal Adjust A	$(A)_{10}$	INH	3721	—	2	—	—	—	—	Δ	Δ	U	Δ
DEC	Decrement Memory	(M) – $01 ⇒ M	IND8, X	01	ff	8	—	—	—	—	Δ	Δ	Δ	—
			IND8, Y	11	ff	8								
			IND8, Z	21	ff	8								
			IND16, X	1701	gggg	8								
			IND16, Y	1711	gggg	8								
			IND16, Z	1721	gggg	8								
			EXT	1731	hh ll	8								
DECA	Decrement A	(A) – $01 ⇒ A	INH	3701	—	2	—	—	—	—	Δ	Δ	Δ	—
DECB	Decrement B	(B) – $01 ⇒ B	INH	3711	—	2	—	—	—	—	Δ	Δ	Δ	—
DECW	Decrement Memory Word	(M : M + 1) – $0001 ⇒ M : M + 1	IND16, X	2701	gggg	8	—	—	—	—	Δ	Δ	Δ	—
			IND16, Y	2711	gggg	8								
			IND16, Z	2721	gggg	8								
			EXT	2731	hh ll	8								
EDIV	Extended Unsigned Divide	(E : D) / (IX) Quotient ⇒ IX Remainder ⇒ D	INH	3728	—	24	—	—	—	—	Δ	Δ	Δ	Δ
EDIVS	Extended Signed Divide	(E : D) / (IX) Quotient ⇒ IX Remainder ⇒ ACCD	INH	3729	—	38	—	—	—	—	Δ	Δ	Δ	Δ
EMUL	Extended Unsigned Multiply	(E) • (D) ⇒ E : D	INH	3725	—	10	—	—	—	—	Δ	Δ	—	Δ
EMULS	Extended Signed Multiply	(E) • (D) ⇒ E : D	INH	3726	—	8	—	—	—	—	Δ	Δ	—	Δ
EORA	Exclusive OR A	(A) ⊕ (M) ⇒ A	IND8, X	44	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	54	ff	6								
			IND8, Z	64	ff	6								
			IMM8	74	ii	2								
			IND16, X	1744	gggg	6								
			IND16, Y	1754	gggg	6								
			IND16, Z	1764	gggg	6								
			EXT	1774	hh ll	6								
			E, X	2744	—	6								
			E, Y	2754	—	6								
			E, Z	2764	—	6								
EORB	Exclusive OR B	(B) ⊕ (M) ⇒ B	IND8, X	C4	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	D4	ff	6								
			IND8, Z	E4	ff	6								
			IMM8	F4	ii	2								
			IND16, X	17C4	gggg	6								
			IND16, Y	17D4	gggg	6								
			IND16, Z	17E4	gggg	6								
			EXT	17F4	hh ll	6								
			E, X	27C4	—	6								
			E, Y	27D4	—	6								
			E, Z	27E4	—	6								
EORD	Exclusive OR D	(D) ⊕ (M : M + 1) ⇒ D	IND8, X	84	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	94	ff	6								
			IND8, Z	A4	ff	6								
			E, X	2784	—	6								
			E, Y	2794	—	6								
			E, Z	27A4	—	6								
			IMM16	37B4	jjkk	4								
			IND16, X	37C4	gggg	6								
			IND16, Y	37D4	gggg	6								
			IND16, Z	37E4	gggg	6								
			EXT	37F4	hhll	6								
EORE	Exclusive OR E	(E) ⊕ (M : M + 1) ⇒ E	IMM16	3734	jj kk	4	—	—	—	—	Δ	Δ	0	—
			IND16, X	3744	gggg	6								
			IND16, Y	3754	gggg	6								
			IND16, Z	3764	gggg	6								
			EXT	3774	hh ll	6								

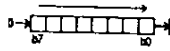
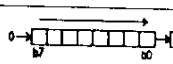
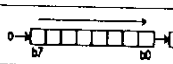
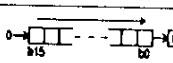
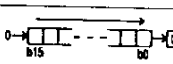
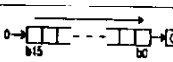
Instruction Set Summary (Continued)

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

Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction		Cycles	Condition Codes							
				Opcode	Operand		S	MV	H	EV	N	Z	V	C
LDAA	Load A	(M) ⇒ A	IND8, X	45	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	55	ff	6								
			IND8, Z	65	ff	6								
			IMM8	75	ii	2								
			IND16, X	1745	gggg	6								
			IND16, Y	1755	gggg	6								
			IND16, Z	1765	gggg	6								
			EXT	1775	hh ll	6								
			E, X	2745	—	6								
			E, Y	2755	—	6								
			E, Z	2765	—	6								
LDAB	Load B	(M) ⇒ B	IND8, X	C5	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	D5	ff	6								
			IND8, Z	E5	ff	6								
			IMM8	F5	ii	2								
			IND16, X	17C5	gggg	6								
			IND16, Y	17D5	gggg	6								
			IND16, Z	17E5	gggg	6								
			EXT	17F5	hh ll	6								
			E, X	27C5	—	6								
			E, Y	27D5	—	6								
			E, Z	27E5	—	6								
LDD	Load D	(M : M + 1) ⇒ D	IND8, X	85	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	95	ff	6								
			IND8, Z	A5	ff	6								
			E, X	2785	—	6								
			E, Y	2795	—	6								
			E, Z	27A5	—	6								
			IMM16	37B5	jj kk	4								
			IND16, X	37C5	gggg	6								
			IND16, Y	37D5	gggg	6								
			IND16, Z	37E5	gggg	6								
			EXT	37F5	hh ll	6								
LDE	Load E	(M : M + 1) ⇒ E	IMM16	3735	jj kk	4	—	—	—	—	Δ	Δ	0	—
			IND16, X	3745	gggg	6								
			IND16, Y	3755	gggg	6								
			IND16, Z	3765	gggg	6								
			EXT	3775	hh ll	6								
LDED	Load Concatenated E and D	(M : M + 1) ⇒ E (M + 2 : M + 3) ⇒ D	EXT	2771	hh ll	8	—	—	—	—	—	—	—	—
LDHI	Initialize H and I	(M : M + 1)X ⇒ H R (M : M + 1)Y ⇒ I R	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								

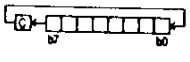
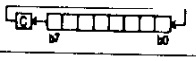
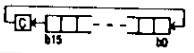
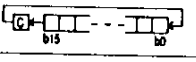
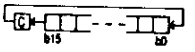
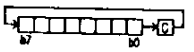
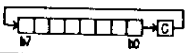
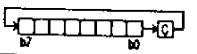
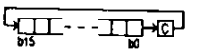
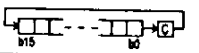
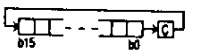
Instruction Set Summary (Continued)

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

Instruction Set Summary (Continued)

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

Instruction Set Summary (Continued)

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

Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction			Condition Codes							
				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 (D) ⇒ M + 2 : M + 3	EXT	2773	hh ll	8	—	—	—	—	—	—	—	—
STS	Store SP	(SP) ⇒ M : M + 1	IND8, X	8F	ff	4	—	—	—	—	Δ	Δ	0	—
			IND8, Y	9F	ff	4								
			IND8, Z	AF	ff	4								
			IND16, X	178F	gggg	6								
			IND16, Y	179F	gggg	6								
			IND16, Z	17AF	gggg	6								
			EXT	17BF	hh ll	6								

Instruction Set Summary (Continued)

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

Instruction Set Summary (Continued)

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

Instruction Set Summary (Concluded)

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

NOTES:

1. CCR[15:4] change according to results of operation. 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
AM	—	Accumulator M
B	—	Accumulator B
CCR	—	Condition code register
D	—	Accumulator D
E	—	Accumulator E
EK	—	Extended addressing extension field
IR	—	MAC multiplicand register
HR	—	MAC multiplier register
IX	—	Index register X
IY	—	Index register Y
IZ	—	Index register Z
K	—	Address extension register
PC	—	Program counter
PK	—	Program counter extension field
SK	—	Stack pointer extension field
SL	—	Multiply and accumulate sign latch
SP	—	Stack pointer
XK	—	Index register X extension field
YK	—	Index register Y extension field
ZK	—	Index register Z extension field
XMSK	—	Modulo addressing index register X mask
YMSK	—	Modulo addressing index register Y mask
S	—	Stop disable control bit
MV	—	AM overflow indicator
H	—	Half carry indicator
EV	—	AM extended overflow indicator
N	—	Negative indicator
Z	—	Zero indicator
V	—	Two's complement overflow indicator
C	—	Carry/borrow indicator
IP	—	Interrupt priority field
SM	—	Saturation mode control bit
PK	—	Program counter extension field
—	—	Bit not affected
Δ	—	Bit changes as specified
0	—	Bit cleared
1	—	Bit set
M	—	Memory location used in operation
R	—	Result of operation
S	—	Source data

+	—	Addition
-	—	Subtraction or negation (two's complement)
*	—	Multiplication
/	—	Division
>	—	Greater
<	—	Less
=	—	Equal
≥	—	Equal or greater
≤	—	Equal or less
≠	—	Not equal

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

•	—	AND
+	—	Inclusive OR (OR)
⊕	—	Exclusive OR (EOR)
$\overline{\text{NOT}}$	—	Complementation
:	—	Concatenation
⇒	—	Transferred
⇔	—	Exchanged
±	—	Sign bit; also used to show tolerance
«	—	Sign extension
%	—	Binary value
$	—	Hexadecimal value

2.8 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.

2.8.1 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.

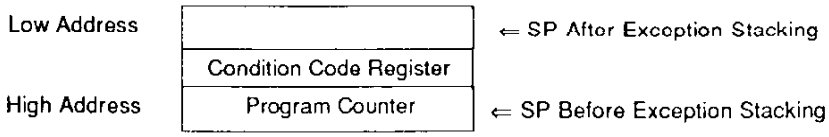
Each vector is assigned an 8-bit number. Vector numbers for some exceptions are generated by external devices; others are supplied by the processor. There is a direct mapping of vector number to vector table address. The CPU16 shifts the vector number left one place (multiplies by two) to convert it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

2.8.2 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.



2.8.3 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 and 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 except the reset vectors contain 16-bit addresses, and the PK field is cleared. Exception handlers must be located within bank 0 or vectors must point to a jump table.

2.8.4 Types of Exceptions

Exceptions can be either internally or externally generated. External exceptions, which are defined as asynchronous, include interrupts, bus errors, breakpoints, and resets. Internal exceptions, which are defined as synchronous, include the software interrupt (SWI) instruction, the background (BGND) instruction, illegal instruction exceptions, and the divide-by-zero exception. 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 reset, 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 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. Because RTI always subtracts $0006 upon return, the stacked PK : PC must be adjusted by the instruction that caused the exception so that execution resumes with the following instruction. $0002 is added to the PK : PC value before it is stacked.

2.8.5 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 highest to lowest. 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 will be 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 exception handler is executed before interrupts are sensed. This permits the exception handler to mask interrupts during execution.

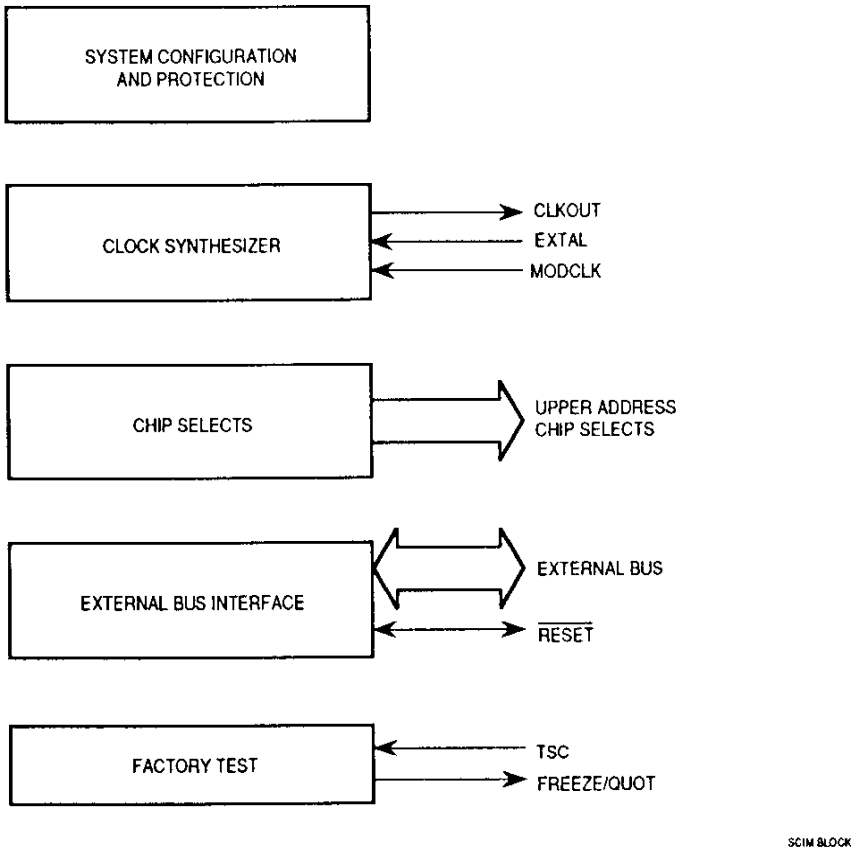
2.8.6 RTI Instruction

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 control system startup, initialization, configuration, and external bus with a minimum of external devices. The SCIM can be configured to operate in 16-bit expanded mode, 8-bit expanded mode, or single-chip mode. Operating mode is determined by the value of the DATA1 and $\overline{\text{BERR}}$ signals coming out of reset. A block diagram of the SCIM follows.



Single-Chip Integration Module Block Diagram

3.1 Overview

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

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

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

The chip-select block provides 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 CONFIGURATION (SCIMCR)	
$YFFA02	FACTORY TEST (SCIMTR)	
$YFFA04	CLOCK SYNTHESIZER CONTROL (SYNCR)	
$YFFA06	UNUSED	RESET STATUS REGISTER (RSR)
$YFFA08	SCIM 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 REGISTER (PORTE0)
$YFFA12	UNUSED	PORT E DATA REGISTER (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 CONTROL (PORTFE)
$YFFA2A	UNUSED	PORT F EDGE-DETECT INTERRUPT VECTOR (PFIVR)
$YFFA2C	UNUSED	PORT F EDGE-DETECT 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 REGISTER (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 logic state of the modmap (MM) bit in the SCIMCR.

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. 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/$\overline{W}$ signal operates normally

1 = R/$\overline{W}$ 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.

3.3 Operating Modes

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

Basic Configuration Options

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

$\overline{BERR}$, $\overline{BKPT}$, 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 ADDR[10:3] = Port B DATA[15:8] = Port G DATA[7:0] = Port H

Many pins on the CPU16, including data and address bus pins, have multiple functions. The reset value for these pins depends on the operating mode in effect. In expanded mode, the values of DATA[11:0] during reset determines the function of these pins. The functions of some pins can be changed by then writing to the appropriate pin assignment register. Data bus pins have internal pull-ups (active only when RESET is asserted) and must be pulled low to achieve the desired alternate configuration. The following tables are a summary of the pin configuration options for each operating mode.

3.3.1 16-Bit Expanded Mode

In 16-bit expanded mode, ($\overline{BERR}$ = 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 Left High)	Alternate Function (Pin Pulled Low)
$\overline{CSBOOT}$	DATA0	$\overline{CSBOOT}$ 16-Bit	$\overline{CSBOOT}$ 8-Bit
$\overline{BR}$/$\overline{CS0}$ FC0/$\overline{CS3}$ FC1/PC1 FC2/$\overline{CS5}$/PC2	DATA2	$\overline{CS0}$ $\overline{CS3}$ FC1 $\overline{CS5}$	BR FC0 FC1 FC2
ADDR19/$\overline{CS6}$/PC3 ADDR20/$\overline{CS7}$/PC4 ADDR21/$\overline{CS8}$/PC5 ADDR22/$\overline{CS9}$/PC6 ADDR23/CS10/ECLK	DATA3 DATA4 DATA5 DATA6 DATA7	$\overline{CS6}$ $\overline{CS[7:6]}$ $\overline{CS[8:6]}$ $\overline{CS[9:6]}$ $\overline{CS[10:6]}$	ADDR19 ADDR[20:19] ADDR[21:19] ADDR[22:19] ADDR[23:19]
$\overline{DSACK0}$/PE0 $\overline{DSACK1}$/PE1 $\overline{AVEC}$/PE2 PE3 $\overline{DS}$/PE4 $\overline{AS}$/PE5 $\overline{SIZ0}$/PE6 $\overline{SIZ1}$/PE7	DATA8	$\overline{DSACK0}$ $\overline{DSACK1}$ $\overline{AVEC}$ PE3 $\overline{DS}$ $\overline{AS}$ $\overline{SIZ0}$ $\overline{SIZ1}$	PE0 PE1 PE2 PE3 PE4 PE5 PE6 PE7
MODCLK/PF0 $\overline{IRQ}$[7:1]/PF[7:1]	DATA9	MODCLK $\overline{IRQ}$[7:1]	PF0 PF[7:1]
$\overline{BGACK}$/$\overline{CSE}$ $\overline{BG}$	DATA10	$\overline{BGACK}$ $\overline{BG}$	$\overline{CSE}$[1] —
DATA11	DATA11	Slave Mode Disabled[2]	Slave Mode Enabled[2]
DATA14	DATA14	EEPROM Normal Mode	EEPROM Stop Mode

NOTES:

1. $\overline{CSE}$ is enabled when DATA10 and DATA1 = 0 during reset.
2. Slave mode used for factory test only.

3.3.2 8-Bit Expanded Mode

In 8-bit expanded mode ($\overline{BERR}$ = 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 Left High)	Alternate Function (Pin Pulled Low)
$\overline{CSBOOT}$	N/A[1]	$\overline{CSBOOT}$ 8-Bit	$\overline{CSBOOT}$ 8-Bit
$\overline{BR}$/$\overline{CS0}$ FC0/$\overline{CS3}$/PC0 FC1/PC1 FC2/$\overline{CS5}$/PC2	N/A[1]	$\overline{CS0}$ $\overline{CS3}$ FC1 $\overline{CS5}$	$\overline{CS0}$ $\overline{CS3}$ FC1 $\overline{CS5}$
ADDR19/$\overline{CS6}$/PC3 ADDR20/$\overline{CS7}$/PC4 ADDR21/$\overline{CS8}$/PC5 ADDR22/$\overline{CS9}$/PC6 ADDR23/CS10/ECLK	N/A[1]	$\overline{CS[10:6]}$	$\overline{CS[10:6]}$
$\overline{DSACK0}$/PE0 $\overline{DSACK1}$/PE1 $\overline{AVEC}$/PE2 PE3 $\overline{DS}$/PE4 $\overline{AS}$/PE5 $\overline{SIZ0}$/PE6 SIZ1/PE7	DATA8	$\overline{DSACK0}$ $\overline{DSACK1}$ $\overline{AVEC}$ PE3 $\overline{DS}$ $\overline{AS}$ $\overline{SIZ0}$ SIZ1	PE0 PE1 PE2 PE3 PE4 PE5 PE6 PE7
MODCLK/PF0 IRQ[7:1]/PF[7:1]	DATA9	MODCLK $\overline{IRQ[7:1]}$	PF0 PF[7:1]
$\overline{BGACK}$/$\overline{CSE}$ $\overline{BG}$	N/A[1]	$\overline{BGACK}$ $\overline{BG}$	$\overline{BGACK}$ $\overline{BG}$
DATA14	DATA14	EEPROM Normal Mode	EEPROM Stop Mode

NOTES:

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

3.3.3 Single-Chip Mode

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

Single-Chip Mode Reset Configuration

Pin(s) Affected	Function
$\overline{CSBOOT}$	$\overline{CSBOOT}$ 8-Bit
ADDR[18:11]	PA[7:0]
ADDR[10:3]	PB[7:0]
BR/$\overline{CS0}$	$\overline{CS0}$
FC0/$\overline{CS3}$/PC0 FC1/PC1 FC2/$\overline{CS5}$/PC2 ADDR19/$\overline{CS6}$/PC3 ADDR20/$\overline{CS7}$/PC4 ADDR21/$\overline{CS8}$/PC5 ADDR22/$\overline{CS9}$/PC6	PC[6:0]
ADDR23/$\overline{CS10}$/ECLK	—
$\overline{DSACK0}$/PE0 $\overline{DSACK1}$/PE1 AVEC/PE2 PE3 $\overline{DS}$/PE4 $\overline{AS}$/PE5 $\overline{SIZ0}$/PE6 $\overline{SIZ1}$/PE7	PE[7:0]
MODCLK/PF0 IRQ[7:1]/PF[7:1]	PF0 PF[7:1]
DATA[15:8]	PG[7:0]
DATA[7:0]	PH[7:0]
$\overline{BGACK}$/$\overline{CSE}$ $\overline{BG}$	$\overline{BGACK}$ $\overline{BG}$

3.4 Emulation Support

The SCIM contains logic that can be used to replace on-chip ports externally. The SCIM also contains special support logic to allow external emulation of internal ROM. This emulation support allows system 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, $\overline{BERR}$ 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 ($\overline{CSE}$) is asserted whenever any of the externally-mapped registers are addressed. The signal is asserted on the falling edge of $\overline{AS}$. The SCIM provides $\overline{DSACK}$ for these accesses, but the data comes from the external data bus. This allows external logic, such as a port replacement unit (PRU) to respond. Accesses to externally-mapped registers require three clock cycles, whether in emulation mode or not.

In devices that contain a masked-ROM module, external ROM emulation is enabled by holding DATA10 and DATA13 low during reset (DATA14 must be held high during reset to enable the ROM module). While ROM emulation mode is enabled, memory chip select signal $\overline{CSM}$ is asserted whenever a valid access to an address assigned to the masked ROM array is made. Because the MC68HC916Y1 has no ROM, the $\overline{CSM}$ function is not used — the CSM pin is driven high whenever the function is selected.

3.5 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	$\overline{MODCLK}$	0	0	0	0	0	0

The system protection control register controls system monitor functions, software watchdog clock prescaling, and bus monitor timing. During normal operation this register can be written only once following power-on or reset, but can be read 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 shows 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

3.5.1 Bus Monitor

The internal bus monitor checks for excessively long response times during normal bus cycles ($\overline{DSACK}$) and during IACK cycles ($\overline{AVEC}$). The monitor asserts $\overline{BERR}$ if response time is excessive.

$\overline{DSACK}$ and $\overline{AVEC}$ 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 $\overline{DSACK}$ response on the external bus unless it initiates the bus cycle. The BME bit in SYPCR enables the internal bus monitor for internal to external bus cycles. If a system contains external bus masters, an external bus monitor must be implemented and the internal to external bus monitor option must be disabled.

3.5.2 Halt Monitor

The halt monitor responds to an assertion of $\overline{HALT}$ 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 HME bit in SYPCR.

3.5.3 Spurious Interrupt Monitor

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

3.5.4 Software Watchdog

SWSR — Software Service Register **$YFFA27**

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

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. SWSR 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 in the order listed prior to timeout, but any number of instructions can be executed between the two writes.

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

When SWT[1:0] are modified, a watchdog service sequence must be performed before the new timeout period takes 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 can be calculated using the following expressions:

$$\text{Timeout Period} = \left(\frac{(128)\,(\text{Divide Ratio})}{\text{EXTAL Frequency}} \right)$$

or

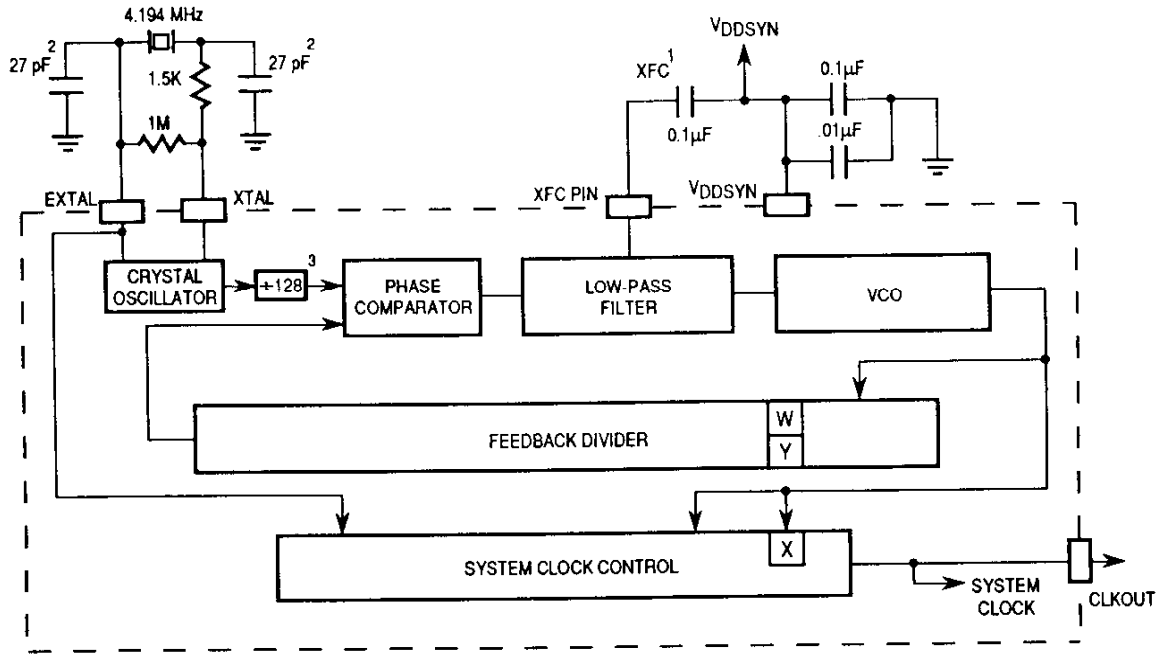
$$\text{Timeout Period} = \left(\frac{\text{System Clock Frequency}}{\text{Divide Ratio}} \right)$$

3.6 System Clock

The system clock in the SCIM 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 KDS041-18 4.194-MHz CRYSTAL. SPECIFIC COMPONENTS MUST BE BASED ON CRYSTAL TYPE. CONTACT CRYSTAL VENDOR FOR EXACT CIRCUIT.
3. 4 MHz DIVIDED TO 32 KHz.

16 SYS CLOCK BLOCK 4MHZ

System Clock Block Diagram

3.6.1 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:

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

3.6.2 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 reference frequency is divided by 128 before being fed to the comparator. 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 + 128. 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. 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:

$$F_{SYSTEM} = \frac{F_{REFERENCE}}{128}[4(Y+1)(2^{2W+X})]$$

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. System frequency is two times reference frequency.

3.6.3 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.

3.6.4 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 **3.11 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.

3.6.5 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	9	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:

$$\text{PIT Period} = \frac{(\text{PIT Modulus})(\text{Prescaler Value})(512)}{\text{EXTAL Frequency}}$$

where

PIT Period = Periodic interrupt timer period
PIT Modulus = Periodic interrupt timer register modulus (PITR[7:0])
EXTAL Frequency = Crystal frequency
Prescaler Value = 512 or 1 depending on the state of the PTP bit in the PITR

3.7 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 uses only 20 of the 24 IMB address lines, ADDR[23:20] are driven to the same state as ADDR19.

The EBI provides dynamic sizing between 8-bit and 16-bit data accesses. It supports byte, word, and long-word transfers. Ports are accessed through the use of asynchronous cycles controlled by the data transfer (SIZ1 and SIZ0) and data size acknowledge pins ($\overline{\text{DSACK1}}$ and $\overline{\text{DSACK0}}$). In fully expanded mode, both 8-bit and 16-bit data ports can be accessed; in partially expanded mode, only 8-bit ports can be accessed. Multiple bus cycles may be required for a transfer to an 8-bit port.

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 **3.11 Chip Selects** for more information.

3.7.1 Bus Control Signals

The CPU16 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 ($\overline{\text{AS}}$) is asserted. The following table shows SIZ0 and SIZ1 encoding. The read/write (R/$\overline{\text{W}}$) signal determines the direction of the transfer during a bus cycle. This signal changes state, when required, at the beginning of a bus cycle, and is valid while $\overline{\text{AS}}$ is asserted. R/$\overline{\text{W}}$ 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

3.7.2 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 $\overline{AS}$ 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

3.7.3 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 $\overline{AS}$ is asserted. Because the CPU16 does not use ADDR[23:20], these lines are driven to the same logic state as ADDR19.

3.7.4 Address Strobe

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

3.7.5 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 $\overline{AS}$ is asserted in a write cycle.

3.7.6 Data Strobe

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

3.7.7 Bus Cycle Termination Signals

During bus cycles, external devices assert the data transfer and size acknowledge signals ($\overline{DSACK1}$ and $\overline{DSACK0}$). During a read cycle, the signals tell the MCU to terminate the bus cycle and to latch data. During a write cycle, the signals indicate that an external device has successfully stored data and that the cycle can end. These signals also indicate to the MCU the size of the port for the bus cycle just completed. (Refer to the discussion of dynamic bus sizing.)

The bus error ($\overline{BERR}$) signal is also a bus cycle termination indicator and can be used in the absence of $\overline{DSACK1}$ and $\overline{DSACK0}$ to indicate a bus error condition. It can also be asserted in conjunction with these signals, provided it meets the appropriate timing requirements. The internal bus monitor can be used to generate the $\overline{BERR}$ signal for internal and internal-to-external transfers. When $\overline{BERR}$ and $\overline{HALT}$ are asserted simultaneously, the CPU16 takes a bus error exception.

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

3.7.8 Data Transfer Mechanism

The MCU architecture supports byte, word, and long-word operands, allowing access to 8- and 16-bit data ports through the use of asynchronous cycles controlled by the data transfer and size acknowledge inputs ($\overline{DSACK1}$ and $\overline{DSACK0}$).

3.7.9 Dynamic Bus Sizing

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

Effect of $\overline{DSACK}$ Signals

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

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

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

The MCU always attempts to transfer the maximum amount of data on all bus cycles. For a word operation, it is assumed that the port is 16 bits wide when the bus cycle begins. Operand bytes are designated as shown in 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

3.7.10 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.

3.7.11 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.

3.7.12 Operand Transfer Cases

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

Operand Alignment

Transfer Case	SIZ1	SIZ0	ADDR0	$\overline{\text{DSACK1}}$	$\overline{\text{DSACK0}}$	DATA [15:8]	DATA [7:0]
Byte to 8-Bit Port	0	1	X	1	0	OP0	(OP0)
Byte to 16-Bit Port (Even)	0	1	0	0	X	OP0	(OP0)
Byte to 16-Bit Port (Odd)	0	1	1	0	X	(OP0)	OP0
Word to 8-Bit Port (Aligned)	1	0	0	1	0	OP0	(OP1)
Word to 8-Bit Port (Misaligned)	1	0	1	1	0	OP0	(OP0)
Word to 16-Bit Port (Aligned)	1	0	0	0	X	OP0	OP1
Word to 16-Bit Port (Misaligned)	1	0	1	0	X	(OP0)	OP0
3 Byte to 8-Bit Port (Aligned)[2]	1	1	0	1	0	OP0	(OP1)
3 Byte to 8-Bit Port (Misaligned)[2]	1	1	1	1	0	OP0	(OP0)
3 Byte to 16-Bit Port (Aligned)[2]	1	1	0	0	X	OP0	OP1
3 Byte to 16-Bit Port (Misaligned)[2]	1	1	1	0	X	(OP0)	OP0
Long Word to 8-Bit Port (Aligned)	0	0	0	1	0	OP0	(OP1)
Long Word to 8-Bit Port (Misaligned)[3]	1	0	1	1	0	OP0	(OP0)
Long Word to 16-Bit Port (Aligned)	0	0	0	0	X	OP0	OP1
Long Word to 16-Bit Port (Misaligned)[3]	1	0	1	0	X	(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.

3.8 Reset

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 $\overline{\text{RESET}}$ 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 $\overline{\text{RESET}}$ is asserted, reset does not occur until the clock starts. Resets are clocked to allow completion of write cycles in progress at the time $\overline{\text{RESET}}$ 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.

15 — 8	7	6	5	4	3			0
NOT USED	EXT	POW	SW	HLT	0	LOC	SYS	TST

The reset status register contains a bit for each reset source in the MCU. A set bit indicates what type of reset has occurred. When multiple reset sources occur at the same time, more than one bit in RSR can be set. The reset status register is updated by the reset control logic when the MCU comes out of reset. This register can be read at any time. A write has no effect.

EXT — External Reset
Reset was caused by an external signal.

POW — Power-Up Reset
Reset was caused by the power-up reset circuit.

SW — Software Watchdog Reset
Reset was caused by the software watchdog circuit.

HLT — Halt Monitor Reset
Reset was caused by the system protection submodule halt monitor.

LOC — Loss of Clock Reset
Reset was caused by loss of clock submodule frequency reference. This reset can only occur if the RSTEN bit in the clock submodule is set and the VCO is enabled.

SYS — System Reset
Reset was caused by a CPU RESET instruction. Since the CPU16 has no RESET instruction, this bit is not used, and always reads zero.

TST — Test Submodule Reset
Reset was caused by the test submodule.

3.8.1 SCIM Reset Mode Selection

The logic states of certain MCU pins during reset determine SCIM operating configuration. Refer to **3.3 Operating Modes** for more information.

3.8.2 MCU Module Pin Function During Reset

As a general rule, module pins default to port functions, and input/output ports are set to input state. This is accomplished by disabling pin functions in the appropriate control registers, and by clearing the appropriate port data direction registers. Refer to individual module sections in this technical summary for more information. 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
	$\overline{BKPT}$/DSCLK	$\overline{BKPT}$/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
MCCI	PMC7/TXDA	Discrete Input
	PMC6/RXDA	Discrete Input
	PMC5/TXDB	Discrete Input
	PMC4/RXDB	Discrete Input
	PMC3/$\overline{SS}$	Discrete Input
	PMC2/SCK	Discrete Input
	PMC1/MOSI	Discrete Input
	PMC0/MISO	Discrete Input
TPU	TP[15:0]	TPU Input

3.8.3 Reset Timing

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

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

If an internal source asserts a reset signal, the reset control logic asserts $\overline{RESET}$ 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 $\overline{RESET}$ 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 $\overline{RESET}$ is released.

3.8.4 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} in order for the MCU to operate. The following discussion assumes that V_{DDSYN} is applied before and during reset. This minimizes crystal start-up time. When V_{DDSYN} is applied at power-on, start-up time is affected by specific crystal parameters and by oscillator circuit design. V_{DD} ramp-up time also affects pin state during reset.

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 loss of clock oscillator 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.

3.8.5 Use of Three State Control Pin

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

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

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

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

3.9 Interrupts

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

The CPU16 provides 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.

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 $\overline{IRQ7}$) from being recognized and processed. When IP contains %000, no interrupt is masked. During exception processing, the IP field is set to the priority of the interrupt being serviced.

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

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

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

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

Interrupt 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.

3.9.1 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 in order 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, and if their requests are at the specified IP level, respond to the cycle. Arbitration between simultaneous requests of the same priority is performed by means of serial contention between module interrupt arbitration (IARB) field bit values.

Each module that can make an interrupt service request, including the 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, and the reset IARB value for all other modules is %0000. Initialization software must assign different IARB values in order to implement an arbitration scheme.

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.

Chip-select logic can also be used to generate internal bus termination signals in response to external interrupt requests. Chip-select addresses match logic functions only after the EBI transfers an interrupt acknowledge cycle to the external address bus following IARB contention. When the CPU acknowledges an interrupt request from an internal module, chip-select logic does not respond to the interrupt acknowledge cycle.

The periodic interrupt timer (PIT) 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 **3.6.5 Periodic Interrupt Timer** for more information.

Each of the interrupt request pins can be configured for edge-detection. When a pin is used for edge-detection, it cannot be used for external interrupt service requests. SCIM edge-detection logic can generate an internal interrupt service request, provided proper preconditions are met. There is only one edge-detection interrupt. By hardware convention, edge-detect interrupt requests are serviced after both PIT and external interrupt requests. Refer to **3.10.3 Port F** for more information.

3.9.2 Interrupt Processing Summary

A summary of the interrupt processing sequence follows. When the sequence begins, a valid interrupt service request has been detected and is pending. Chip-select logic can be used to generate $\overline{\text{DSACK}}$ or $\overline{\text{AVEC}}$ termination signals for external interrupt requests, but the processing sequence is not affected.

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 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 DSACK signals appropriate to the access. The CPU16 acquires the vector number.
   2. The AVEC signal is asserted (the signal can be asserted by the dominant interrupt source or the pin can be tied low), and the CPU16 generates an autovector number corresponding to interrupt priority.
   3. The bus monitor asserts 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.

3.10 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[7:1]/MODCLK	All
G	DATA[15:8]	Single Chip
H	DATA[7:0]	Single Chip, 8-Bit Expanded

Access to the port A, B, E, G, and H data and data direction registers, and the port E pin assignment register requires 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 emulator mode is enabled, accesses to ports A, B, E, G, and H data and data direction registers and port E pin assignment register are ignored, and can be replaced with external logic, such as a Motorola port replacement unit (PRU). Port F registers 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.

3.10.1 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. The port A and B registers can be read or written at any time the MCU is not in emulation 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.

3.10.2 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

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.

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 (Expanded, Single chip):	DATA8 0	DATA8 0	DATA8 0	DATA8 0	DATA8 0	DATA8 0	DATA8 0	DATA8 0

The bits in PEPAR control the function of each port E pin. Setting a bit defines the corresponding pin to be a bus control signal, with the function shown in the following table. Clearing a bit defines the corresponding pin as an I/O pin, controlled by PORTE and DDRE.

Port E Pin Assignments

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

* When PEPA3 is set, the PE3 pin goes to logic level one. The CPU16 does not support the control function for this pin.

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

3.10.3 Port F

Port F pins can be configured as level-sensitive interrupt request inputs, edge-detect inputs, or discrete inputs/outputs. The edge-detection logic can make an interrupt service request when the specified edge is detected. In order to enable the edge-detect interrupt request, an interrupt priority level must be specified by writing a value to the port F interrupt level register (PFLVR). The edge-detect interrupt has the lowest hardware priority in the SCIM — both PIT and external interrupt requests have higher priority.

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 an internal data latch. 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 emulation 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	PFPA7	PFPA6	PFPA5	PFPA4	PFPA3	PFPA2	PFPA1	PFPA0
RESET (Expanded, Single chip):	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9
	0	0	0	0	0	0	0	0

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

PFPAR Function

PFPAR Field	Pin Affected	PFPA Bits	Pin Function
PFPA3	PF[7:6]/$\overline{IRQ}$[7:6]	00	I/O pin
PFPA2	PF[5:4]/$\overline{IRQ}$[5:4]	01	Rising edge detection
PFPA1	PF[3:2]/$\overline{IRQ}$[3:2]	10	Falling edge detection
PFPA0	PF[1:0]/$\overline{IRQ1}$, MODCLK*	11	Interrupt request

*MODCLK signal is only recognized during reset.

PORTFE — Port F Edge-Detect Flag Register **$YFFA29**

	15	14	13	12	11	10	9	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 when an edge is detected. PORTFE bits remain set until cleared, regardless of the subsequent state of the corresponding pin. 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	14	13	12	11	10	9	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 to service interrupts generated by the port F edge-detection logic. Program PFIVR[7:0] to the appropriate interrupt vector number. Refer to **3.9 Interrupts** for more information.

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

	15	14	13	12	11	10	9	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 port F edge-detect interrupt requests. The reset value is $00, indicating that interrupts are disabled. The port F edge-detect interrupt has the lowest priority of SCIM interrupt sources — both PIT and external interrupt service requests take precedence over an edge-detection interrupt.

3.10.4 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.

3.10.5 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

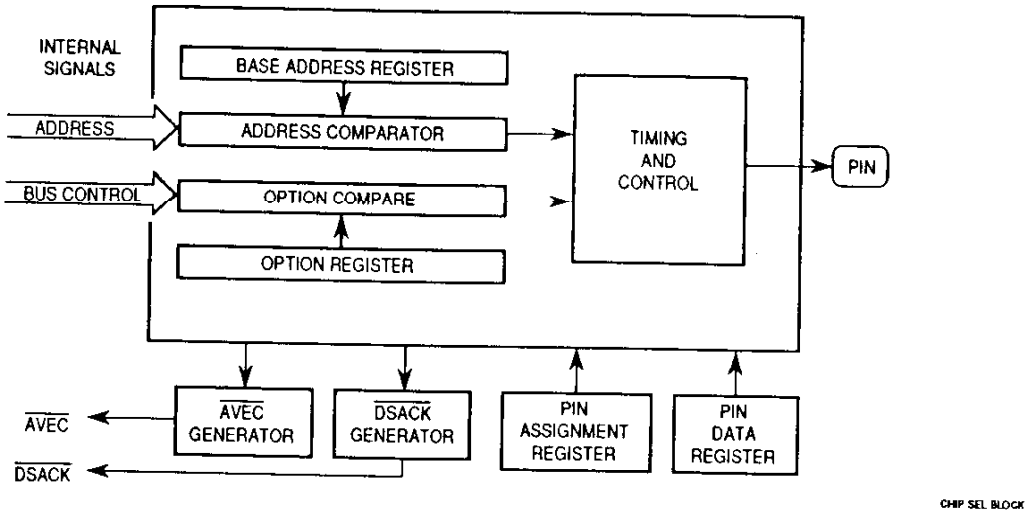
The bits in this register control the direction of the port pin drivers when pins are configured as I/O. Setting a bit configures the corresponding pin as an output. Clearing a bit configures the corresponding pin as an input.

3.11 Chip Selects

Typical microcontrollers require additional hardware to provide external chip-select signals. The MC68HC916Y1 includes nine programmable chip select circuits that can provide 2 to 13 clock cycle access to external memory and peripherals. Two additional chip selects, $\overline{CSE}$ and $\overline{CSM}$, 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 $\overline{DSACK}$ and $\overline{AVEC}$ signals internally. A single $\overline{DSACK}$ 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

Because initialization software usually resides in a peripheral memory device controlled by chip-select circuits, a CSBOOT register provides default reset values to support bootstrap operation.

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 all modes, the boot ROM select signal is automatically asserted out of reset. 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 functions for chip-select pins.

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
$\overline{CSBOOT}$	$\overline{CSBOOT}$	—
$\overline{CS0}$	$\overline{BR}$	—
—	$\overline{BG}$	—
$\overline{CSE}$	$\overline{BGACK}$	—
$\overline{CS3}$	FC0	PC0
—	FC1	PC1
$\overline{CS5}$	FC2	PC2
$\overline{CS6}$	ADDR19	PC3
$\overline{CS7}$	ADDR20	PC4
$\overline{CS8}$	ADDR21	PC5
$\overline{CS9}$	ADDR22	PC6
$\overline{CS10}$	ADDR23	ECLK

3.11.1 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 emulator 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 emulator chip select signal ($\overline{CSE}$) 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 **3.4 Emulation Support** for further information.

3.11.2 Chip Select Registers

Pin assignment registers (CSPAR) determine the 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.

3.11.3 Pin Assignment Registers

The pin assignment registers contain pairs of bits that determine the functions of chip select pins. Alternate functions of the associated pins are shown in the pin assignment tables. Reset value depends on the operating mode.

In the following register diagrams, reset values are shown in the following order: single-chip modes, 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]		CSPA0[5]		CSPA0[4]		CSPA0[3]		CSPA0[2]		CSPA0[1]		$\overline{\text{CSBOOT}}$	
RESET:															
0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1
0	0	1	0	0	1	1	0	0	1	0	1	1	0	1	0
0	0	DATA2	1	0	1	DATA2	1	$\overline{\text{DATA10}}$	1	$\overline{\text{DATA10}}$	1	DATA2	1	1	DATA0

CSPAR0[15:14] — Not used

These bits always read zero; write has no effect.

CSPAR011 — Not used

CSPAR010 determines whether pin is FC1 or a discrete output.

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

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

CSPAR1[15:10] — Not used

These bits always read zero; write has no effect.

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 Assignments

CSPAR0 Field	CSPAR0 Signal	Alternate Signal
CSPA0[6]	$\overline{CS5}$	FC2
CSPA0[5]	—	FC1
CSPA0[4]	$\overline{CS3}$	FC0
CSPA0[3]	$\overline{CSE}$	$\overline{BGACK}$
CSPA0[2]	—	$\overline{BG}$
CSPA0[1]	$\overline{CS0}$	$\overline{BR}$
$\overline{CSBOOT}$	$\overline{CSBOOT}$	—

CSPAR1 Pin Assignments

CSPAR1 Field	CSPAR1 Signal	Alternate Signal
CSPA1[4]	$\overline{CS10}$*	ADDR23
CSPA1[3]	$\overline{CS9}$	ADDR22
CSPA1[2]	$\overline{CS8}$	ADDR21
CSPA1[1]	$\overline{CS7}$	ADDR20
CSPA1[0]	$\overline{CS6}$	ADDR19

*Clearing both CSPA1[4] select bits enables the M6800 bus clock (ECLK) on ADDR23.

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

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

When a pin is programmed for discrete output or alternate function, internal chip select logic is inhibited.

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.

3.11.4 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	1	0
ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	BLKSZ		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

CSBAR0 – CSBAR10 — Chip Select Base Address Registers **$YFFA4C–$YFFA74**

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

The CPU16 drives ADDR[23:20] to the same logic state as ADDR19. ADDR[23:20] must match ADDR19 for the chip select to be active.

ADDR[15:3] — 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.

Because ADDR[23:20] are driven to the same logic state as ADDR19, maximum block size is 512 Kbytes — 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]

3.11.5 Option Registers

The option registers contain eight fields that determine the timing of and conditions for assertion of chip select signals. These fields 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/$\overline{W}$		STRB	$\overline{DSACK}$				SPACE		IPL				$\overline{AVEC}$
RESET:																
0	1	1	1	1	0	1	1	0	1	1	1	0	0	0	0	

CSOR0 – CSOR10 — Chip Select Option Registers **$YFFA4E–$YFFA76**

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

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

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 $\overline{AS}$ or $\overline{DS}$.

The DSACK field is not used in synchronous mode, as 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/$\overline{W}$ — Read/Write

This field causes a chip select to be asserted only for a read, only for a write, or for both read and write.

The following table shows the options.

R/$\overline{W}$	Description
00	Reserved
01	Read Only
10	Write Only
11	Read/Write

STRB — Address Strobe/Data Strobe

0 = Address strobe

1 = Data strobe

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

$\overline{\text{DSACK}}$ — Data Strobe Acknowledge

This field specifies the source of $\overline{\text{DSACK}}$ in asynchronous mode. It also allows the user to adjust bus timing with internal $\overline{\text{DSACK}}$ generation by controlling the number of wait states that are inserted. This function optimizes 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. The fast termination encoding is used for two-cycle access to external memory.

$\overline{\text{DSACK}}$	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 $\overline{\text{DSACK}}$

SPACE — Address Space

This option field is used to select an address space for 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 signal 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 encoding only affects chip-select assertion — it has no effect on interrupt recognition by the CPU.

IPL	Space = 00	Space = 01, 10, 11
000	All priority levels	Data or Program
001	Priority level 1	Data
010	Priority level 2	Program
011	Priority level 3	Reserved
100	Priority level 4	Reserved
101	Priority level 5	Data
110	Priority level 6	Program
111	Priority level 7	Reserved

$\overline{\text{AVEC}}$ — 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 in conjunction with a chip select pin.

If the chip select is configured to trigger on an interrupt acknowledge cycle (SPACE = %00) and the AVEC bit is set, 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:		0	1	1	1	1	1	1	1

The state of bits in PORTC determines the state of pins programmed as port C discrete outputs. When a pin is assigned as a discrete output, the value in this register appears at the output. FC[6:0] correspond to pins $\overline{\text{CS}}$[9:3]. This is a read/write register. Bit 7 is not used. Writing to this bit has no effect and it always reads zero.

3.11.6 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 **3.8.1 SCIM Reset Mode Selection** and to the discussion of the CSPAR0 and CSPAR1 registers for more information.

The $\overline{\text{CSBOOT}}$ assignment field in CSPAR0 is configured differently. The MSB, bit 1 of CSPAR0, is always one. This enables the $\overline{\text{CSBOOT}}$ signal to select a boot ROM containing initialization firmware. The LSB value, determined by the logic level of DATA0 during reset, selects boot ROM port size. When DATA0 is held low, port size is 8 bits. When internal connections pull the LSB high, port size is 16 bits.

After reset, the MCU fetches initialization vectors from addresses $0000 to $0006 in bank 0 of program space. To support bootstrap operation from reset, the bits in the base address field in CSBARBT have a reset value of zero. A ROM device containing vectors located at these addresses can be enabled by the CSBOOT signal after a reset. The block size field in CSBARBT has a reset value of 512 Kbytes.

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. The following table shows the reset values in the base and option registers for CSBOOT.

Chip Select Reset Values

Field	Reset Value
Base Address	$0000 0000
BLKSZ	512 Kbyte
MODE	Asynchronous Mode
BYTE	Both Bytes
R/W	Read/Write
STRB	AS
DSACK	13 Wait States
SPACE	Supervisor/User
IPL	All
AVEC	External Interrupt Vector

3.11.7 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 — Test Module Master Shift Register A **$YFFA30**

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

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

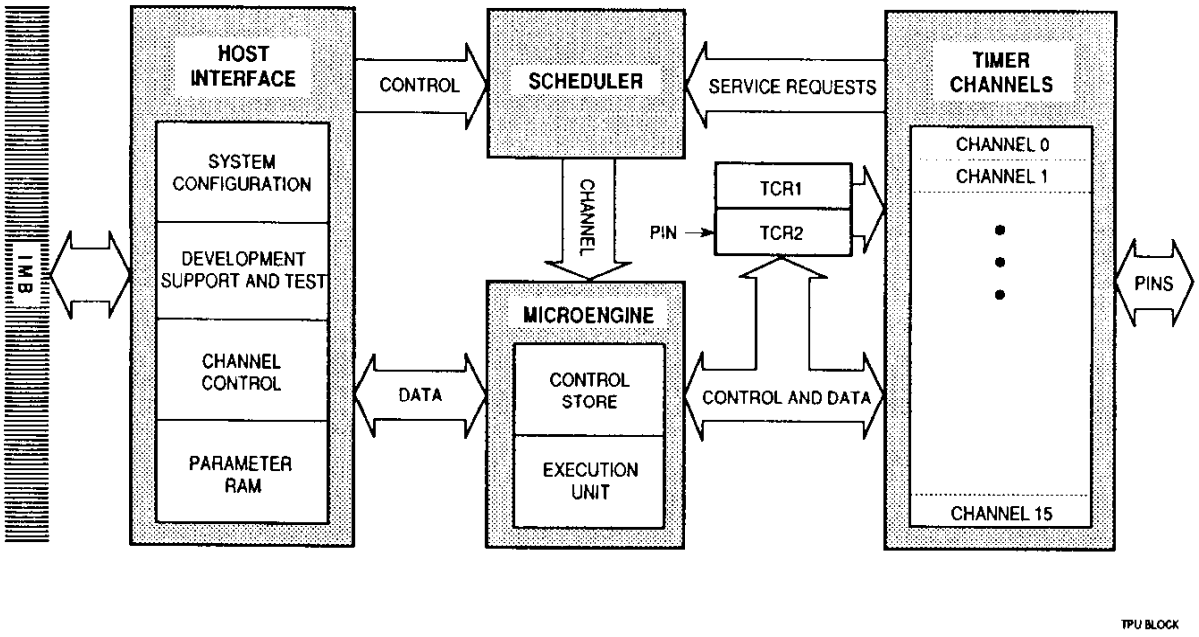
TSTRC — Test Module Repetition Count **$YFFA36**

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

DREG — Test Module Distributed Register **$YFFA3A**

4 Time Processor Unit

The time processor unit (TPU) is an intelligent, semi-autonomous microcontroller designed for timing control. The TPU operates simultaneously with the CPU; it processes ROM instructions, schedules tasks, performs input and output, and accesses shared data without CPU intervention. Consequently, setup and service time for each timer event are minimized. The figure below is a simplified block diagram of the TPU.



TPU Block Diagram

4.1 Overview

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

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

4.2 Programmer's Model

The TPU control register address map occupies 512 bytes. Unused registers within the 512-byte address space return zeros when read.

TPU Address Map

Address	15 8	7 0
$YFFE00	TPU MODULE CONFIGURATION REGISTER (TPUMCR)	
$YFFE02	TEST CONFIGURATION REGISTER (TCR)	
$YFFE04	DEVELOPMENT SUPPORT CONTROL REGISTER (DSCR)	
$YFFE06	DEVELOPMENT SUPPORT STATUS REGISTER (DSSR)	
$YFFE08	TPU INTERRUPT CONFIGURATION REGISTER (TICR)	
$YFFE0A	CHANNEL INTERRUPT ENABLE REGISTER (CIER)	
$YFFE0C	CHANNEL FUNCTION SELECTION REGISTER 0 (CFSR0)	
$YFFE0E	CHANNEL FUNCTION SELECTION REGISTER 1 (CFSR1)	
$YFFE10	CHANNEL FUNCTION SELECTION REGISTER 2 (CFSR2)	
$YFFE12	CHANNEL FUNCTION SELECTION REGISTER 3 (CFSR3)	
$YFFE14	HOST SEQUENCE REGISTER 0 (HSQR0)	
$YFFE16	HOST SEQUENCE REGISTER 1 (HSQR1)	
$YFFE18	HOST SERVICE REQUEST REGISTER 0 (HSRR0)	
$YFFE1A	HOST SERVICE REQUEST REGISTER 1 (HSRR1)	
$YFFE1C	CHANNEL PRIORITY REGISTER 0 (CPR0)	
$YFFE1E	CHANNEL PRIORITY REGISTER 1 (CPR1)	
$YFFE20	CHANNEL INTERRUPT STATUS REGISTER (CISR)	
$YFFE22	LINK REGISTER (LR)	
$YFFE24	SERVICE GRANT LATCH REGISTER (SGLR)	
$YFFE26	DECODED CHANNEL NUMBER REGISTER (DCNR)	

Y = M111, where M represents the logic state of the MODMAP bit in the SCIMCR. In M68HC16 devices, M must equal 1.

4.3 TPU Components

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

4.3.1 Time Bases

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

4.3.2 Timer Channels

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

4.3.3 Scheduler

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

4.3.4 Microengine

The microengine is composed of a control store and an execution unit. Control-store ROM holds the microcode for each factory-masked time function. When assigned to a channel by the scheduler, the execution unit executes microcode for a function assigned to that channel by the host CPU. Microcode can also be executed from the TPURAM module instead of the control store. The TPURAM module allows emulation and development of custom TPU microcode without the generation of a microcode ROM mask. Refer to **4.5 Emulation Support** for more information.

4.3.5 Host Interface

Host interface registers allow communication between the host CPU and the TPU, both before and during execution of a time function. The registers are accessible from the IMB through the TPU bus interface unit.

4.3.6 Parameter RAM

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

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

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

PSC — Forces the output level of the pin.
PAC — For input capture, PAC specifies the edge transition to be detected. For output comparison, PAC specifies the logic level to be output when a match occurs.
TBS — Specifies channel direction (input or output) and assigns a time base to the input capture and output compare functions of the channel.

TPU Parameter RAM Address Map

Channel Number	Base Address	Parameter Address							
		0	1	2	3	4	5	6	7
0	$YFFFF—	00	02	04	06	08	0A	—	—
1	$YFFFF—	10	12	14	16	18	1A	—	—
2	$YFFFF—	20	22	24	26	28	2A	—	—
3	$YFFFF—	30	32	34	36	38	3A	—	—
4	$YFFFF—	40	42	44	46	48	4A	—	—
5	$YFFFF—	50	52	54	56	58	5A	—	—
6	$YFFFF—	60	62	64	66	68	6A	—	—
7	$YFFFF—	70	72	74	76	78	7A	—	—
8	$YFFFF—	80	82	84	86	88	8A	—	—
9	$YFFFF—	90	92	94	96	98	9A	—	—
10	$YFFFF—	A0	A2	A4	A6	A8	AA	—	—
11	$YFFFF—	B0	B2	B4	B6	B8	BA	—	—
12	$YFFFF—	C0	C2	C4	C6	C8	CA	—	—
13	$YFFFF—	D0	D2	D4	D6	D8	DA	—	—
14	$YFFFF—	E0	E2	E4	E6	E8	EA	EC	EE
15	$YFFFF—	F0	F2	F4	F6	F8	FA	FC	FE

— = Not Implemented

Y = M111, where M represents the logic state of the MODMAP bit in the SCIMCR. In M68HC16 devices, M must equal 1.

4.4 TPU Operation

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

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

4.4.1 Event Timing

Match and capture events are handled by independent channel hardware. This provides an event accuracy of one time-base clock period, regardless of the number of channels that are active. An event normally causes a channel to request service. However, before an event can be serviced, any pending previous requests must be serviced. The time needed to respond to and service an event is determined by the number of channels requesting service, the relative priorities of the channels requesting service, and the microcode execution time of the active functions. Worst-case event service time (latency) determines TPU performance in a given application. Latency can be closely estimated — see Motorola *TPU Reference Manual* (TPURM/AD) for more information.

4.4.2 Channel Orthogonality

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

4.4.3 Interchannel Communication

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

4.4.4 Programmable Channel Service Priority

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

4.4.5 Coherency

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

4.5 Emulation Support

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

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

4.6 Time Functions

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

4.6.1 Discrete Input/Output

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

4.6.2 Input Capture/Input Transition Counter

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

4.6.3 Output Compare

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

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

$$\text{OFFSET} = \text{PERIOD} * \text{RATIO}$$

where RATIO is a parameter supplied by the user.

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

4.6.4 Pulse-Width Modulation

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

4.6.5 Synchronized Pulse-Width Modulation

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

4.6.6 Period Measurement with Additional Transition Detect

This function and the following function are used primarily in toothed-wheel speed-sensing applications, such as monitoring rotational speed of an engine. The period measurement with additional transition detect (PMA) function allows for a special-purpose 23-bit period measurement. It can detect the occurrence of an additional transition (caused by an extra tooth on the sensed wheel) indicated by a period measurement that is less than a programmable ratio of the previous period measurement. Once detected, this condition can be counted and compared to a programmable number of additional transitions detected before TCR2 is reset to $FFFF. Alternatively, a byte at an address specified by a channel parameter can be read and used as a flag. A nonzero value of the flag indicates that TCR2 is to be reset to $FFFF once the next additional transition is detected.

4.6.7 Period Measurement with Missing Transition Detect

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

4.6.8 Position-Synchronized Pulse Generator

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

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

4.6.9 Stepper Motor

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

The time period between steps (P) is defined as

$$P(r) = K1 - K2 * r$$

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

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

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

4.6.10 Period/Pulse-Width Accumulator

The period/pulse-width accumulator (PPWA) algorithm accumulates a 16-bit or 24-bit sum of either the period or the pulse width of an input signal over a programmable number of periods or pulses (from 1 to 255). After an accumulation period, the algorithm can generate a link to a sequential block of up to eight channels. The user specifies a starting channel of the block and number of channels within the block. Generation of links depends on the mode of operation.

Any channel can be used to measure an accumulated number of periods of an input signal. A maximum of 24 bits can be used for the accumulation parameter. From 1 to 255 period measurements can be made and summed with the previous measurement(s) before the TPU interrupts the CPU, allowing instantaneous or average frequency measurement, and the latest complete accumulation (over the programmed number of periods).

The pulse width (high-time portion) of an input signal can be measured (up to 24 bits) and added to a previous measurement over a programmable number of periods (1 to 255). This provides an instantaneous or average pulse-width measurement capability, allowing the latest complete accumulation (over the specified number of periods) to always be available in a parameter.

By using the output compare function in conjunction with PPWA, an output signal can be generated that is proportional to a specified input signal. The ratio of the input and output frequency is programmable. One or more output signals with different frequencies, yet proportional and synchronized to a single input signal, can be generated on separate channels.

4.7 TPU Registers

The TPU memory map contains three groups of registers:

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

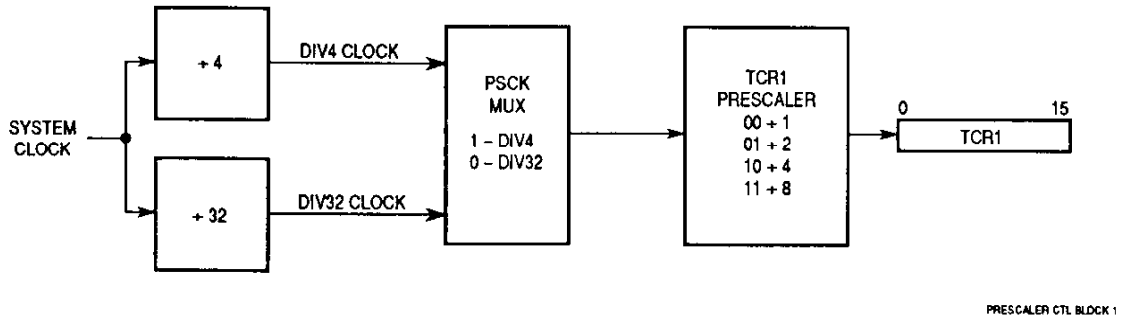
4.7.1 System Configuration Registers

TPUMCR — TPU Module Configuration Register **$YFFE00**

15	14	13	12	11	10	9	8	7	6	5	4	3	0
STOP	TCR1P		TCR2P		EMU	T2CG	STF	SUPV	PSCK	0	0	IARB	
RESET:													
0	0	0	0	0	0	0	0	1	0	0	0	0	0 0 0 0

STOP — Stop Bit
0 = TPU operating normally
1 = Internal clocks shut down

TCR1P — Timer Count Register 1 Prescaler Control
TCR1 is clocked from the output of a prescaler. The prescaler's input is the internal TPU system clock divided by either 4 or 32, depending on the value of the PSCK bit. The prescaler divides this input by 1, 2, 4, or 8. Channels using TCR1 have the capability to resolve down to the TPU system clock divided by 4.



Prescaler Control 1

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

TCR2P — Timer Count Register 2 Prescaler Control

TCR2 is clocked from the output of a prescaler. If T2CG = 0, the input to the TCR2 prescaler is the external TCR2 clock source. If T2CG = 1, the input is the TPU system clock divided by eight. The TCR2P field specifies the value of the prescaler: 1, 2, 4, or 8. Channels using TCR2 have the capability to resolve down to the TPU system clock divided by 8. The following table is a summary of prescaler output.

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

EMU — Emulation Control

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

0 = TPU and TPURAM not in emulation mode

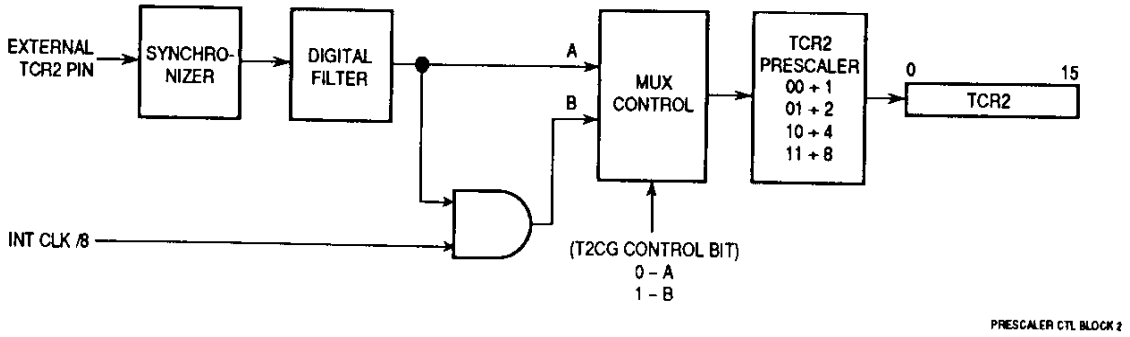
1 = TPU and TPURAM in emulation mode

T2CG — TCR2 Clock/Gate Control

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

0 = TCR2 pin used as clock source for TCR2

1 = TCR2 pin used as gate of DIV8 clock for TCR2



Prescaler Control 2

STF — Stop Flag

0 = TPU operating

1 = TPU stopped (STOP bit has been asserted)

SUPV — Supervisor Data Space

0 = Assignable registers are unrestricted (FC2 is ignored)
1 = Assignable registers are restricted (FC2 is decoded)

PSCK — Prescaler Clock

0 = System clock/32 is input to TCR1 prescaler
1 = System clock/4 is input to TCR1 prescaler

IARB — Interrupt Arbitration Number

This field contains the arbitration number of the TPU that is used to arbitrate for the intermodule bus when two or more modules or peripherals have an interrupt on the same priority level.

TICR — TPU Interrupt Configuration Register **$YFFE08**

15	11	10	8	7	4	3	0
NOT USED		CIRL		CIBV		NOT USED	
RESET:		0	0	0	0	0	0	0	

CIRL — Channel Interrupt Request Level

The interrupt request level for all channels is specified by this 3-bit encoded field. Level seven for this field indicates a nonmaskable interrupt; level zero indicates that all channel interrupts are disabled.

CIBV — Channel Interrupt Base Vector

The TPU is assigned 16 unique interrupt vector numbers, one vector number for each channel. The CIBV field specifies the most significant nibble of all 16 TPU channel interrupt vector numbers. The lower nibble of the TPU interrupt vector number is determined by the channel number on which the interrupt occurs.

4.7.2 Channel Control Registers

CIER — Channel Interrupt Enable Register **$YFFE0A**

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

CH[15:0] — Channel Interrupt Enable/Disable

0 = Channel interrupts disabled
1 = Channel interrupts enabled

CISR — Channel Interrupt Status Register **$YFFE20**

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

CH[15:0] — Channel Interrupt Status Bit
0 = Channel interrupt not asserted
1 = Channel interrupt asserted

CFSR0 — Channel Function Select Register 0 **$YFFE0C**

15	12	11	8	7	4	3	0
CHANNEL15		CHANNEL14		CHANNEL13		CHANNEL12	
RESET:							
0	0	0	0	0	0	0	0

CFSR1 — Channel Function Select Register 1 **$YFFE0E**

15	12	11	8	7	4	3	0
CHANNEL11		CHANNEL10		CHANNEL9		CHANNEL8	
RESET:							
0	0	0	0	0	0	0	0

CFSR2 — Channel Function Select Register 2 **$YFFE10**

15	12	11	8	7	4	3	0
CHANNEL7		CHANNEL6		CHANNEL5		CHANNEL4	
RESET:							
0	0	0	0	0	0	0	0

CFSR3 — Channel Function Select Register 3 **$YFFE12**

15	12	11	8	7	4	3	0
CHANNEL3		CHANNEL2		CHANNEL1		CHANNEL0	
RESET:							
0	0	0	0	0	0	0	0

CHANNEL[15:0] — Encoded Time Function for each Channel
Encoded 4-bit fields in the channel function select registers specify one of 16 time functions to be executed on the corresponding channel. Encodings for predefined functions are found in the table **Host Service Request and Sequence Codes**.

HSQR0 — Host Sequence Register 0 **$YFFE14**

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

HSQR1 — Host Sequence Register 1 **$YFFE16**

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

CH[15:0] — Encoded Host Sequence

The host sequence field selects the mode of operation for the time function selected on a given channel. The meaning of the host sequence bits depends on the time function specified. Refer to the table, **Host Service Request and Sequence Codes**, which is a summary of the host sequence and host service request bits for each predefined time function.

HSRR0 — Host Service Request Register 0 **$YFFE18**

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

HSRR1 — Host Service Request Register 1 **$YFFE1A**

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

CH[15:0] — Encoded Type of Host Service

The host service request field selects the type of host service request for the time function selected on a given channel. The meaning of the host service request bits depends on the time function specified. The table, **Host Service Request and Sequence Codes**, is a summary of the host sequence and host service request bits for each predefined time function.

A host service request field cleared to %00 signals the host that service is completed by the microengine on that channel. The host can request service on a channel by writing the corresponding host service request field to one of three nonzero states. The CPU should monitor the host service request register until the TPU clears the service request to %00 before the CPU changes any parameters or issues a new service request to the channel.

Host Service Request and Sequence Codes

Function Name	Function Code	Host Service Request Code	Host Sequence Code*
DIO Discrete Input/Output	$8	1= Force Output High	0 = Trans Mode — Record Pin on Transition
		2 = Force Output Low	0 = Trans Mode — Record Pin on Transition
		3 = Initialization, Input Spec.	0 = Trans Mode — Record Pin on Transition
		3 = Initialization, Periodic Input	1 = Match Mode — Record Pin at Match_Rate
		3 = Update Status Parameter	2 = Record Pin State on HSR 11
ITC Input Capture/ Input Transition Counter	$A	0 = None 1 = Initialization 2 = (Not Implemented) 3 = (Not Implemented)	0 = No Link, Single Mode 1 = No Link, Continuous Mode 2 = Link, Single Mode 3 = Link, Continuous Mode
OC Output Compare	$E	0 = None 1 = Host-Initiated Pulse Mode 2 = (Not Implemented) 3 = Continuous Pulse Mode	0 = Execute All Functions 1 = Execute All Functions 2 = Only Update TCRn Parameters 3 = Only Update TCRn Parameters
PWM Pulse-Width Modulation	$9	0 = None 1 = Immediate Update Request 2 = Initialization 3 = (Not Implemented)	(None Implemented)
SPWM Synchronized Pulse-Width Modulation	$7	0 = None 1 = (Not Implemented) 2 = Initialization 3 = Immediate Update Request	0 = Mode 0 1 = Mode 1 2 = Mode 2 3 = (Not Implemented)
PMA/PMM Period Measurement with Additional/Missing Transition Detect	$B	0 = None 1 = Initialization 2 = (Not Implemented) 3 = (Not Implemented)	0 = PMA Bank Mode 1 = PMA Count Mode 2 = PMM Bank Mode 3 = PMM Count Mode
PSP Position-Synchronized Pulse Generator	$C	0 = None 1 = Immediate Update Request 2 = Initialization 3 = Force Change	0 = Pulse Width Set by Angle 1 = Pulse Width Set by Time 2 = Pulse Width Set by Angle 3 = Pulse Width Set by Time
SM Stepper Motor	$D	0 = None 1 = None 2 = Initialization 3 = Step Request	(None Implemented)
PPWA Period/Pulse-Width Accumulator	$F	0 = None 1 = (Not Implemented) 2 = Initialization 3 = (Not Implemented)	0 = 24-Bit Period 1 = 16-Bit Period + Link 2 = 24-Bit Pulse Width 3 = 16-Bit Pulse Width + Link

CPR0 — Channel Priority Register 0 **$YFFE1C**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH 15		CH 14		CH13		CH 12		CH 11		CH 10		CH 9		CH 8	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CPR1 — Channel Priority Register 1 **$YFFE1E**

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

CH[15:0] — Encoded One of Three Channel Priority Levels

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

4.7.3 Development Support and Test Registers

These registers are used for custom microcode development or for factory test. Describing the use of the registers is beyond the scope of this technical summary. Register names and addresses are given for reference only. Please refer to the *TPU Reference Manual* (TPURM/AD) for more information.

DSCR — Development Support Control Register **$YFFE04**

DSSR — Development Support Status Register **$YFFE06**

LR — Link Register **$YFFE22**

SGLR — Service Grant Latch Register **$YFFE24**

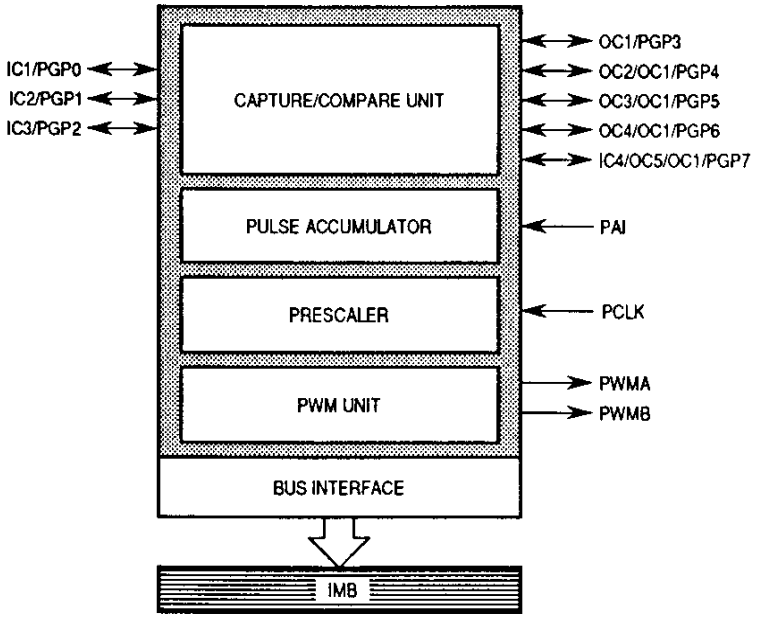
DCNR — Decoded Channel Number Register **$YFFE26**

TCR — Test Configuration Register **$YFFE02**

The TCR is used for factory test of the MCU.

5 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 Diagram

5.1 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.

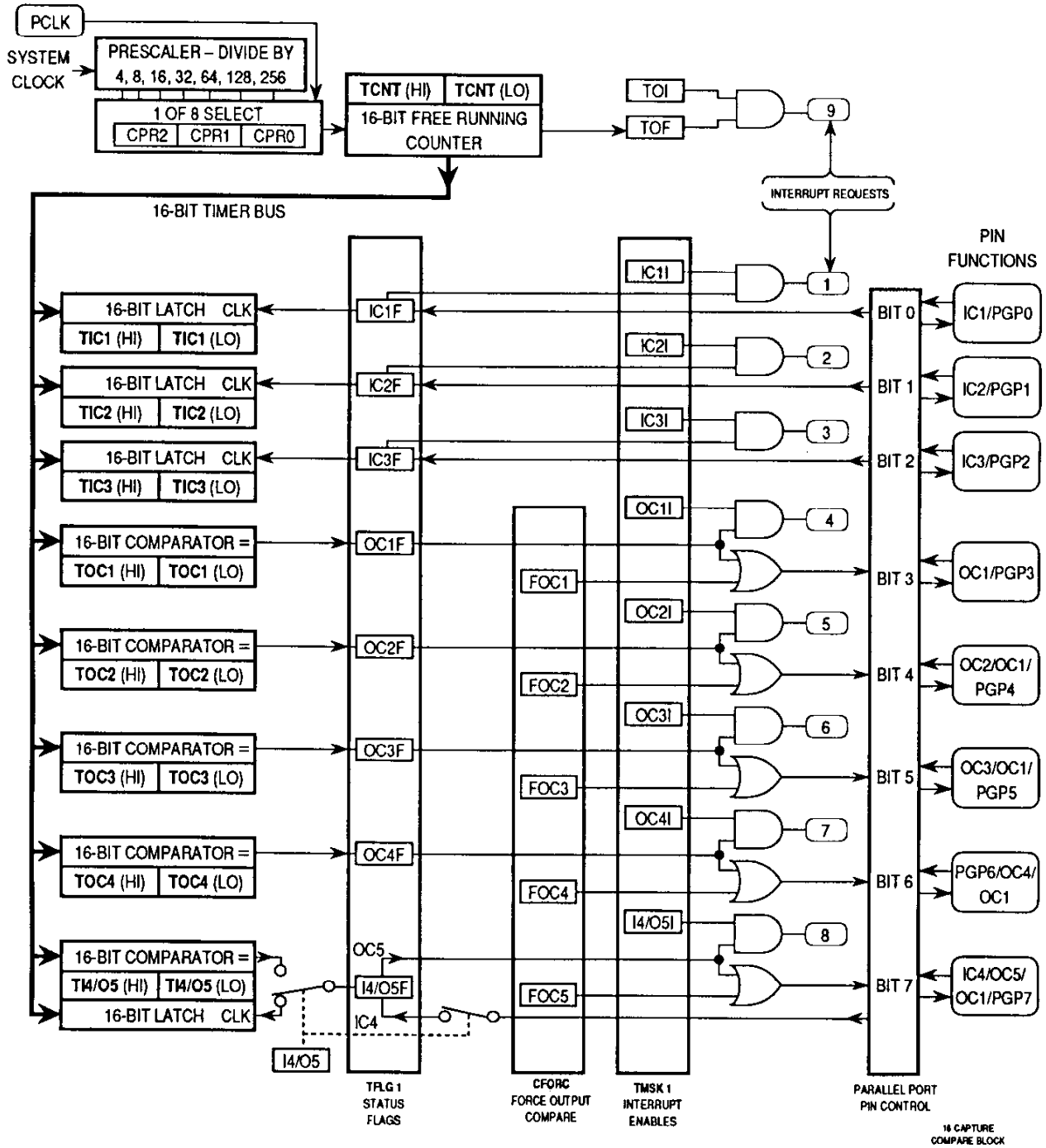
GPT Address Map

Address	15 8	7 0
$YFF900	GPT MODULE CONFIGURATION (GPTMCR)	
$YFF902	(RESERVED FOR TEST)	
$YFF904	INTERRUPT CONFIGURATION (ICR)	
$YFFE06	PGP DATA DIRECTION (DDRGP)	PGP DATA (PORTGP)
$YFF908	OC1 ACTION MASK (OC1M)	OC1 ACTION DATA (OC1D)
$YFF90A	TIMER COUNTER (TCNT)	
$YFF90C	PA CONTROL (PACTL)	PA COUNTER (PACNT)
$YFF90E	INPUT CAPTURE 1 (TIC1)	
$YFF910	INPUT CAPTURE 2 (TIC2)	
$YFF912	INPUT CAPTURE 3 (TIC3)	
$YFF914	OUTPUT COMPARE 1 (TOC1)	
$YFF916	OUTPUT COMPARE 2 (TOC2)	
$YFF918	OUTPUT COMPARE 3 (TOC3)	
$YFF91A	OUTPUT COMPARE 4 (TOC4)	
$YFF91C	INPUT CAPTURE 4/OUTPUT COMPARE 5 (TI4/O5)	
$YFF91E	TIMER CONTROL 1 (TCTL1)	TIMER CONTROL 2 (TCTL2)
$YFF920	TIMER MASK 1 (TMSK1)	TIMER MASK 2 (TMSK2)
$YFF922	TIMER FLAG 1 (TFLG1)	TIMER FLAG 2 (TFLG2)
$YFF924	FORCE COMPARE (CFORC)	PWM CONTROL C (PWMC)
$YFF926	PWM CONTROL A (PWMA)	PWM CONTROL B (PWMB)
$YFF928	PWM COUNT (PWMCNT)	
$YFF92A	PWMA BUFFER (PWMBUFA)	PWMB BUFFER (PWMBUFB)
$YFF92C	GPT PRESCALER (PRESCL)	
$YFF92E–$YFF93F	RESERVED	

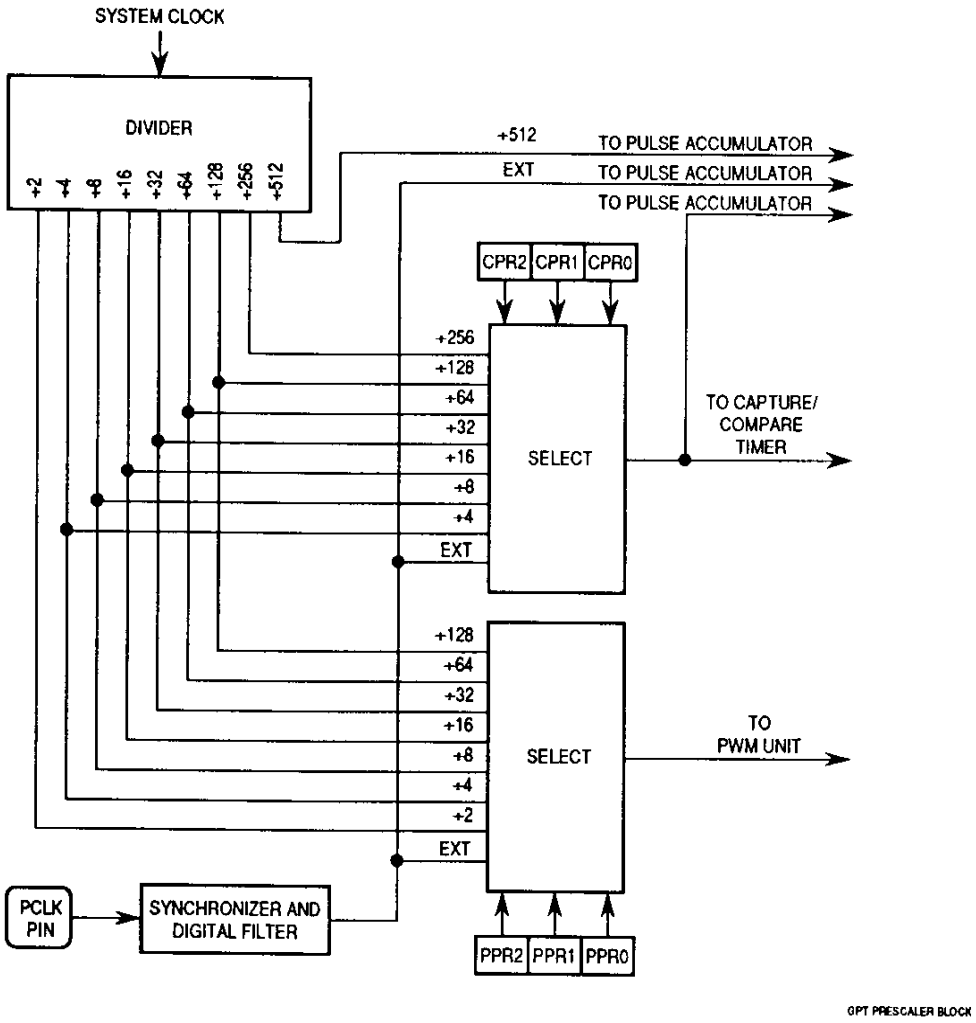
Y = M111, where M is the logic state of the modmap (MM) bit in SCIMCR.

5.2 Capture/Compare Unit

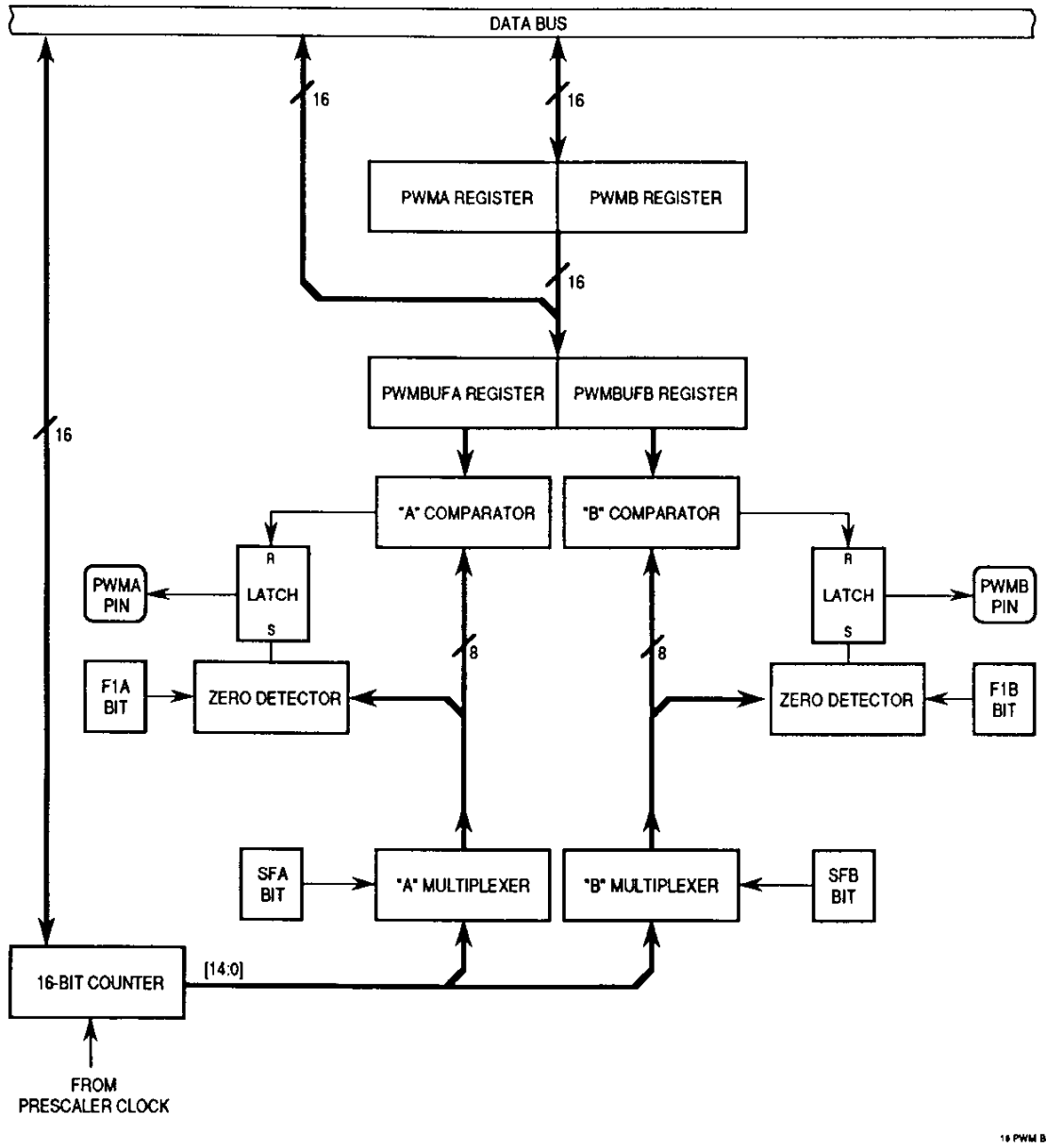
The capture/compare unit features three input capture channels, four output compare channels, and one input capture/output compare channel (function selected by control register). These channels share a 16-bit free-running counter (TCNT), which derives its clock from seven stages of a 9-stage prescaler or from external clock input PCLK. This section, which is similar to the timer found on the MC68HC11F1, also contains one pulse accumulator channel. The pulse accumulator logic includes its own 8-bit counter and can operate in either event counting mode or gated time accumulation mode. Refer to the following block diagrams of the GPT timer and prescaler.



GPT Timer Block Diagram



Prescaler Block Diagram



PWM Unit Block Diagram

5.3 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 can 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 compare/capture unit) or by the clock input pin, PCLK.

5.4 GPT Registers

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

15	14	13	12	11	10	9	8	7	6	5	4	3			0
STOP	FRZ1	FRZ0	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 configuring the GPT.

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

FRZ1 — Not implemented at this time

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 — Interrupt Arbitration Identification

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 returns 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
IPA				0	IPL			IVBA			0	0	0	0
RESET:														
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IPA — Interrupt Priority Adjust

Specifies which GPT interrupt source is given highest internal priority

IPL — Interrupt Priority Level

Specifies the priority level of interrupts generated by the GPT.

IVBA — Interrupt Vector Base Address

Most significant nibble of interrupt vector numbers generated by the GPT.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DDGP7	DDGP6	DDGP5	DDGP4	DDGP3	DDGP2	DDGP1	DDGP0	PGP7	PGP6	PGP5	PGP4	PGP3	PGP2	PGP1	PGP0
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

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	0

All OC outputs can be controlled by the action of OC1. OC1M contains a mask that determines which pins are affected. 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
8-bit read/write counter used for external event counting or gated time accumulation.

TIC[1:3] — 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.

TOC[1:4] — 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	0

TMSK1 enables OC and IC interrupts. TMSK2 controls pulse accumulator interrupts and TCNT functions.

I4/O5I — Input Capture 4/Output Compare 5 Interrupt Enable
0 = IC4/OC5 interrupt disabled
1 = IC4/OC5 interrupt requested when I4/O5F flag in TFLG1 is set

OCI[4:1] — Output Compare Interrupt Enable
0 = OC interrupt disabled
1 = OC interrupt requested when OC flag set
OCI[4:1] correspond to OC[4:1].

ICI[3:1] — Input Capture Interrupt Enable
0 = IC interrupt disabled
1 = IC interrupt requested when IC flag set
ICI[3:1] correspond to IC[3:1].

TOI — Timer Overflow Interrupt Enable
0 = Timer overflow interrupt disabled
1 = Interrupt requested when TOF flag is set

PAOVI — Pulse Accumulator Overflow Interrupt Enable
0 = Pulse accumulator overflow interrupt disabled
1 = Interrupt requested when PAOVF flag is set

PAII — Pulse Accumulator Input Interrupt Enable
0 = Pulse accumulator interrupt disabled
1 = Interrupt requested when PAIF flag is set

CPROUT — Compare/Capture Unit Clock Output Enable
0 = Normal operation for OC1 pin
1 = TCNT clock driven out OC1 pin

CPR[2:0] — Timer Prescaler/PCLK Select Field
This field selects one of seven prescaler taps or PCLK to be TCNT input.

CPR[2:0]	System Clock Divide-by Factor
000	4
001	8
010	16
011	32
100	64
101	128
110	256
111	PCLK

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

These registers show condition flags that correspond to various GPT events. If the corresponding interrupt enable bit in TMSK1/TMSK2 is set, an interrupt occurs.

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, PAIF is set at the end of the timed period.

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

Setting a bit in CFORC causes a specific output on OC or PWM pins. PWMC sets PWM operating conditions.

FOC[5:1] — Force Output Compare

0 = Has no meaning

1 = Causes pin action programmed for corresponding OC pin, but the OC flag is not set.

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

FPWMA — Force PWMA Value

0 = Normal PWMA operation

1 = The value of F1A is driven out on the PWMA pin, regardless of the state of PPROUT.

FPWMB — Force PWMB Value

0 = Normal PWMB operation

1 = The value of F1B is driven out on the PWMB pin.

PPROUT — PWM Clock Output Enable

0 = Normal PWM operation on PMWA

1 = TCNT clock driven out PWMA pin

PPR[2:0] — PWM Prescaler/PCLK Select

This field selects one of seven prescaler taps, or PCLK, to be PWMCNT input.

PPR[2:0]	System Clock Divide-by Factor
000	2
001	4
010	8
011	16
100	32
101	64
110	128
111	PCLK

SFA — PWMA Slow/Fast Select

0 = PWMA period is 256 PWMCNT increments long.

1 = PWMA period is 32768 PWMCNT increments long.

SFB — PWMB Slow/Fast Select

0 = PWMB period is 256 PWMCNT increments long.

1 = PWMB period is 32768 PWMCNT increments long.

The following table shows the effects of SF settings on PWM frequency (16.78-MHz system clock).

PPR[2:0]	Prescaler Tap	SFA/B = 0	SFA/B = 1
000	Div 2 = 8.39 MHz	32.8 kHz	256 Hz
001	Div 4 = 4.19 MHz	16.4 kHz	128 Hz
010	Div 8 = 2.10 MHz	8.19 kHz	64.0 Hz
011	Div 16 = 1.05 MHz	4.09 kHz	32.0 Hz
100	Div 32 = 524 kHz	2.05 kHz	16.0 Hz
101	Div 64 = 262 kHz	1.02 kHz	8.0 Hz
110	Div 128 = 131 kHz	512 Hz	4.0 Hz
111	PCLK	PCLK/256	PCLK/32768

F1A — Force Logic Level One on PWMA

0 = Force logic level zero output on PWMA pin

1 = Force logic level one output on PWMA pin

F1B — Force Logic Level One on PWMB

0 = Force logic level zero output on PWMB pin

1 = Force logic level one output on PWMB pin

PWMA/PWMB — PWM Control 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 that 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**

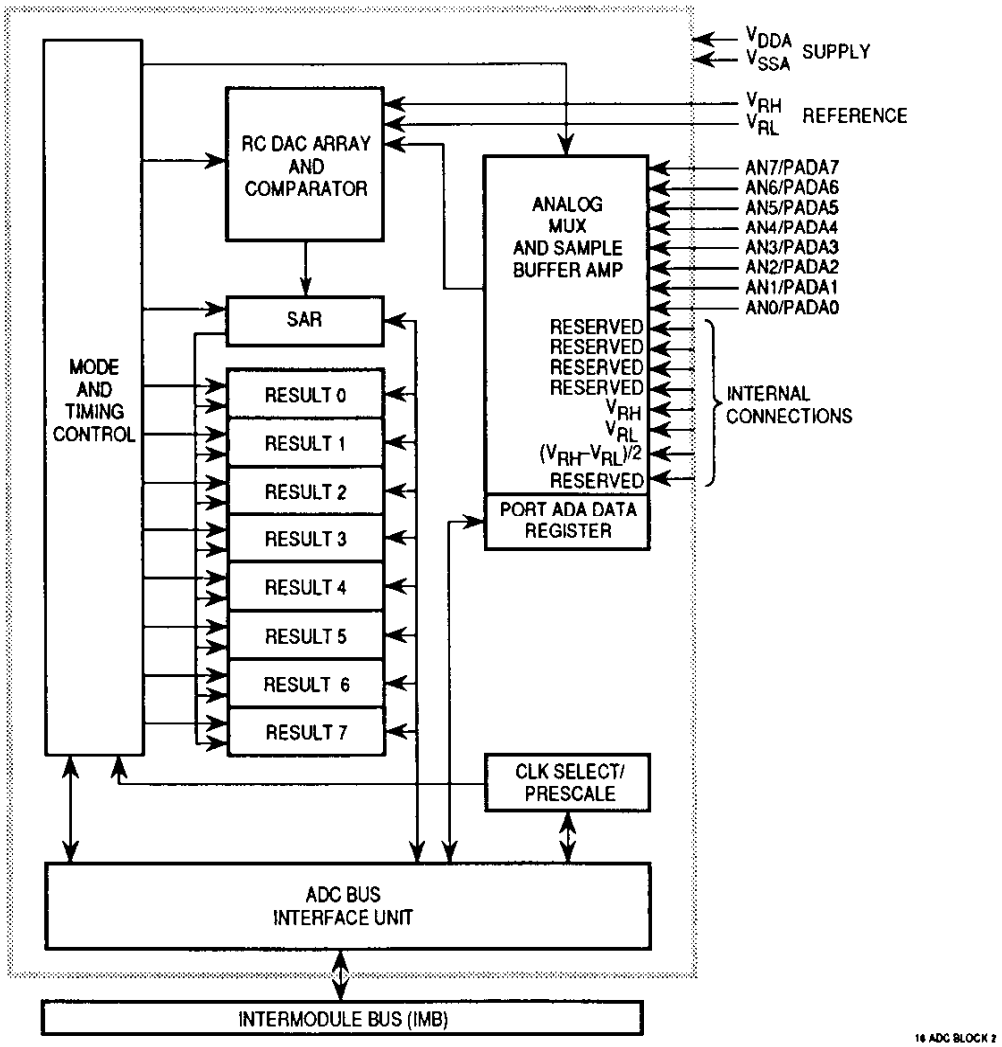
These read-only registers contain values associated with the duty cycles of the corresponding PWM. Reset state is $0000.

PRESCL — GPT Prescaler **$YFF92C**

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

6 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.



Analog-to-Digital Converter Block Diagram

6.1 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.

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 logic state of the modmap (MM) bit in the SCIMCR

6.2 Analog Subsystem

The analog front end consists of a multiplexer, a buffer amplifier, a resistor-capacitor (RC) 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 RC array. The resistor capacitor 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.

6.3 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.

6.4 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.

6.5 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 aborts any conversion in progress. STOP is set to logic level one at reset, and can 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 ADA 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							

PADA[7:0] — Port ADA Data

A read of PADA[7:0] returns the logic level of the port A pins. When the input is outside the defined levels, the read is indeterminate. Use of a port A pin for digital input does not preclude its 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 to 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 aborts it and resets 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 through [CD:CA])

1 = Sequential conversion of a block of four or eight channels (block selected through [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 summarizes 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	RSLT[0:3]
0	0	0	0	1	AN1	RSLT[0:3]
0	0	0	1	0	AN2	RSLT[0:3]
0	0	0	1	1	AN3	RSLT[0:3]
0	0	1	0	0	AN4	RSLT[0:3]
0	0	1	0	1	AN5	RSLT[0:3]
0	0	1	1	0	AN6	RSLT[0:3]
0	0	1	1	1	AN7	RSLT[0:3]
0	1	0	0	0	Reserved	RSLT[0:3]
0	1	0	0	1	Reserved	RSLT[0:3]
0	1	0	1	0	Reserved	RSLT[0:3]
0	1	0	1	1	Reserved	RSLT[0:3]
0	1	1	0	0	V_{RH}	RSLT[0:3]
0	1	1	0	1	V_{RL}	RSLT[0:3]
0	1	1	1	0	$(V_{RH} - V_{RL}) / 2$	RSLT[0:3]
0	1	1	1	1	Test/Reserved	RSLT[0:3]
1	0	0	0	0	AN0	RSLT[0:7]
1	0	0	0	1	AN1	RSLT[0:7]
1	0	0	1	0	AN2	RSLT[0:7]
1	0	0	1	1	AN3	RSLT[0:7]
1	0	1	0	0	AN4	RSLT[0:7]
1	0	1	0	1	AN5	RSLT[0:7]
1	0	1	1	0	AN6	RSLT[0:7]
1	0	1	1	1	AN7	RSLT[0:7]
1	1	0	0	0	Reserved	RSLT[0:7]
1	1	0	0	1	Reserved	RSLT[0:7]
1	1	0	1	0	Reserved	RSLT[0:7]
1	1	0	1	1	Reserved	RSLT[0:7]
1	1	1	0	0	V_{RH}	RSLT[0:7]
1	1	1	0	1	V_{RL}	RSLT[0:7]
1	1	1	1	0	$(V_{RH} - V_{RL}) / 2$	RSLT[0:7]
1	1	1	1	1	Test/Reserved	RSLT[0:7]

The following table is a summary of the operation of S8CM and [CD:CA] when MULT is set (multi-channel 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

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

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 the result register is read.

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

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

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

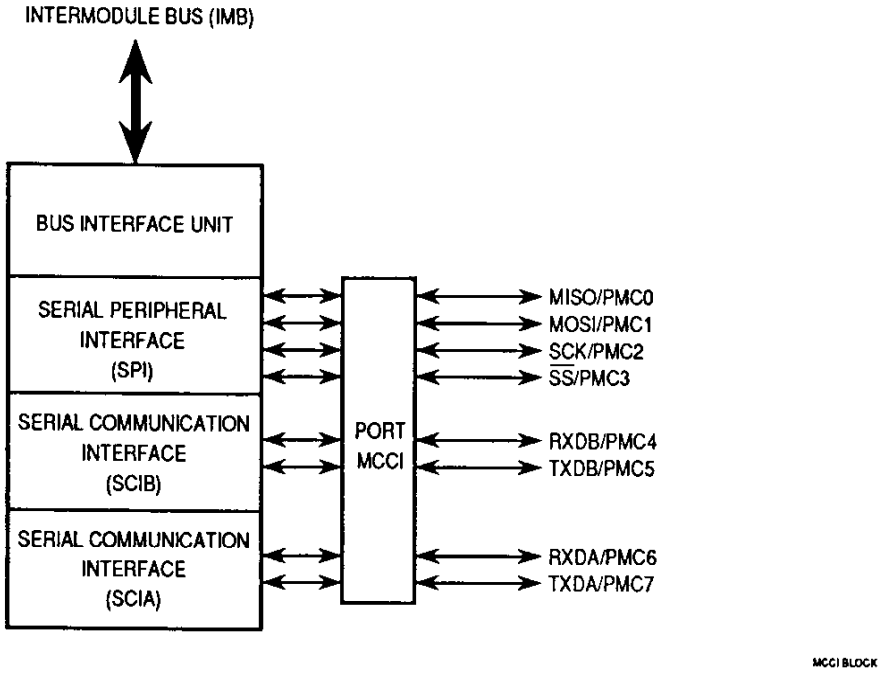
Conversion result is signed left-justified data. Bits [15:6] are used for 10-bit resolution, bits [15:8] are used for 8-bit conversion (bits [7:6] are zero). Although the ADC is unipolar, it is assumed that the zero point is halfway between low and high reference when this format is used. For positive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

7 Multichannel Communication Interface

The MCCI contains three serial interfaces: two serial communication interfaces (SCI) and a serial peripheral interface (SPI). Refer to the following block diagram of the MCCI.



MCCI Block Diagram

7.1 Overview

The two SCI interfaces in the MCCI provide serial communication via a standard nonreturn to zero (NRZ) mark/space format. Either SCI can operate in full or half-duplex mode. There are separate transmitter and receiver enable bits and dual data buffers for each SCI. A modulus-type baud rate generator provides rates from 64 to 524 kbaud (with a 16.78-MHz system clock). Word length of either eight or nine bits is software selectable. Optional parity generation and detection provide either even or odd parity check capability. Advanced error detection circuitry catches glitches of up to 1/16 of a bit time in duration. Wakeup functions allow the CPU to run uninterrupted until meaningful data is available.

The SPI provides easy peripheral expansion or interprocessor communication through a full-duplex, synchronous, three-line bus: data in, data out, and a serial clock. The SPI is compatible with SPI interfaces found in other Motorola devices, but contains enhanced operational features, such as programmable shift direction.

MCCI pins can also be configured for use in 8-bit general-purpose I/O port MC.

MCCI Address Map

Address	15 8	7 0
$YFFC00	MCCI MODULE CONFIGURATION REGISTER (MMCR)	
$YFFC02	MCCI TEST REGISTER (MTEST)	
$YFFC04	SCI INTERRUPT REGISTER (ILSCI)	SCI INTERRUPT VECTOR REGISTER (MIVR)
$YFFC06	SPI INTERRUPT REGISTER (ILSPI)	RESERVED
$YFFC08	RESERVED	PORTMC PIN ASSIGNMENT REGISTER (PMCPAR)
$YFFC0A	RESERVED	PORTMC DATA DIRECTION REGISTER (DDRMC)
$YFFC0C	RESERVED	PORTMC DATA REGISTER (PORTMC)
$YFFC0E	RESERVED	MCCI PORT PIN STATE REGISTER (PORTMCP)
$YFFC10–$YFFC16	RESERVED	
$YFFC18	SCIA CONTROL REGISTER 0 (SCCR0A)	
$YFFC1A	SCIA CONTROL REGISTER 1 (SCCR1A)	
$YFFC1C	SCIA STATUS REGISTER (SCSRA)	
$YFFC1E	SCIA DATA REGISTER (SCDRA)	
$YFFC20–$YFFC26	RESERVED	
$YFFC28	SCIB CONTROL REGISTER 0 (SCCR0B)	
$YFFC2A	SCIB CONTROL REGISTER 1 (SCCR1B)	
$YFFC2C	SCIB STATUS REGISTER (SCSRB)	
$YFFC2E	SCIB DATA REGISTER (SCDRB)	
$YFFC30–$YFFC36	RESERVED	
$YFFC38	SPI CONTROL REGISTER (SPCR)	
$YFFC3A	RESERVED	
$YFFC3C	SPI STATUS REGISTER (SPSR)	
$YFFC3E	SPI DATA REGISTER (SPDR)	

Y = M111, where M is the logic state of the modmap (MM) bit in the SCIMCR

7.2 MCCI Registers

MCCI registers are divided into four categories: MCCI global registers, MCCI pin control registers, SCI registers, and SPI registers. SPI and SCI registers are defined in separate sections later. Writes to unimplemented register bits have no meaning or effect, and reads from unimplemented bits always return a logic zero value.

The modmap bit of the single-chip integration module configuration register (SCIMCR) defines the most significant bit (ADDR23) of the address, shown in each register diagram as **Y**. This bit, concatenated with the rest of the address given, forms the absolute address of each register. The CPU16 drives ADDR[23:20] to the same logic state as ADDR19, and Y must equal $F. Refer to **3 Single-Chip Integration Module** for more information about how the state of MM affects the system.

7.2.1 MCCI Global Registers

Global registers contain parameters used to interface the MCCI with the rest of the MCU. Parameters in global registers affect both the SPI and the SCI as well as the MCCI as a whole.

MMCR — MCCI Configuration Register **$YFFC00**

15	14	8	7	6	4	3	0
STOP	NOT USED		SUPV	NOT USED		IARB	
RESET:							
0			1			0 0 0 0	

STOP — Stop Enable

0 = Normal MCCI clock operation

1 = MCCI clock operation stopped

STOP places the MCCI into a low power state by disabling the system clock in most parts of the module. MMCR is the only register guaranteed to be readable while STOP is asserted. STOP can be negated by the CPU and by reset.

SUPV — Supervisor/Unrestricted

0 = Unrestricted access

1 = Supervisor access

In systems with controlled access levels, SUPV places assignable registers in either supervisor-only data space or unrestricted data space. All MCCI registers reside in supervisor-only space. Because the CPU16 operates only in supervisor mode, SUPV has no meaning.

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.

MTEST — MCCI Test Register **$YFFC02**

MTEST is used in conjunction with SCIM test functions during factory test of the MCCI. Accesses to MTEST must be made while the MCU is in test mode.

ILSCI/MIVR — SCI Interrupt Request Level Register/MCCI Interrupt Vector Register **$YFFC04**

15	14	13	11	10	8	7	2	1	0
0	0	ILSCIB		ILSCIA		MIVR		*	*
RESET:									
0	0	0	0	0	0	0	0	1	1

*Supplied by MCCI

ILSCI determines the priority level of interrupts requested by each SCI. Separate fields hold interrupt priority values for SCIA and SCIB. Priority determines which interrupt is serviced first when two or more modules or external peripherals request an interrupt simultaneously.

ILSCIA, ILSCIB — Interrupt Level for SCIA, SCIB

ILSCIA and ILSCIB determine the priority levels of SCIA and SCIB interrupts, respectively. This field must contain a value between $1 (lowest priority) and $7 (highest priority) for interrupts to be recognized.

MIVR — MCCI Interrupt Vector Register

MIVR determines which vector the CPU uses to service an MCCI interrupt after it is acknowledged. At reset, MIVR is initialized to $0F, which corresponds to the uninitialized interrupt vector in the exception vector table. MIVR must be programmed to one of the user-defined vectors ($40–$FF) during initialization of the MCCI in order for interrupts to be serviced.

MCCI interrupt vectors are adjacent to one another in the exception vector table. MIVR[7:2] are the same for all three interfaces. The MCCI provides the values for MIVR[1:0] according to the source of the interrupt (%00 for SCIA, %01 for SCIB, and %10 for the SPI). Writes to MIVR[1:0] have no meaning or effect. Reads of MIVR[1:0] return a value of %11.

ILSPI — SPI Interrupt Level Register **$YFFC06**

15	14	13	12	11	10	9	8	7	0
0	0	ILSPI			0	0	0	RESERVED	
RESET:									
0	0	0	0	0	0	0	0		

ILSPI determines the priority of interrupts requested by the SPI. The ILSPI field must contain a value between $1 (lowest priority) and $7 (highest priority) for interrupts to be recognized. If ILSPI, ILSCIA, and ILSCIB are the same, simultaneous interrupt requests are recognized in SPI, SCIA, SCIB priority.

7.2.2 MCCI Pin Control Registers

MCCI pin control registers determine the use of eight MCU pins. Although these pins are used by the serial subsystems, any pin can alternately be assigned for use in a general-purpose parallel port. The MCCI pin assignment register (PMCPAR) determines whether pins are assigned to the SPI or to the parallel port. Clearing a bit assigns the corresponding pin to the port; setting a bit assigns the pin to the SPI. PMCPAR does not affect operation of the SCI submodule.

The MCCI data direction register (DDRMC) determines whether pins are inputs or outputs. Clearing a bit makes the corresponding pin an input; setting a bit makes the pin an output. DDRMC affects both SPI function and I/O function. DDRMC determines the direction of SCI TXD pins only when an SCI transmitter is disabled. When an SCI transmitter is enabled, the TXD pin is an output.

MCCI port data register PORTMC latches I/O data; MCCI pin state register PORTMCP allows pin state to be read regardless of data direction configuration.

PORTMC — MCCI Port Data Register **$YFFC0C**

15	8	7	6	5	4	3	2	1	0
RESERVED		PMC7	PMC6	PMC5	PMC4	PMC3	PMC2	PMC1	PMC0

Writes to PORTMC are stored in an internal data latch. If any bit of PORTMC is configured as discrete output, the latched value is driven onto the corresponding pin. Reads of PORTMC return the value of the pin only if the pin is configured as a discrete input. Otherwise, the value read is the latched value. To avoid driving undefined data, first write a byte to PORTMC, then configure DDRMC.

PORTMCP — MCCI Port Pin State Register **$YFFC0E**

15	8	7	6	5	4	3	2	1	0
RESERVED		PMCP7	PMCP6	PMCP5	PMCP4	PMCP3	PMCP2	PMCP1	PMCP0

Reads of PORTMCP always return the state of the pins regardless of whether the pins are configured as input or output. Writes to PORTMCP have no effect.

PMCPAR — MCCI Pin Assignment Register **$YFFC08**

15	8	7	6	5	4	3	2	1	0
RESERVED		0	0	0	0	PMCPA3	0	PMCPA1	PMCPA0
RESET:		0	0	0	0	0	0	0	0

PMCPAR determines which of the SPI pins, with the exception of the SCK pin (the state of which is determined by the SPI enable bit), are used by the SPI submodule, and which pins are available for general-purpose I/O. Clearing a bit in PMCPAR assigns SPI pins for use as general-purpose I/O; setting a bit assigns the pin to the SPI. SPI pins designated by PMCPAR as general-purpose I/O are controlled only by DDRMC and PORTMC; the SPI has no effect on these pins. PMCPAR does not affect the operation of the SCI submodule.

DDRMC — Port MC Data Direction Register **$YFFC0B**

15	8	7	6	5	4	3	2	1	0
RESERVED		DDM7	DDM6	DDM5	DDM4	DDM3	DDM2	DDM1	DDM0
RESET:		0	0	0	0	0	0	0	0

DDRMC determines whether a general-purpose I/O pin is an input or an output. During reset, all MCCI pins are configured as general-purpose inputs. Clearing a bit makes the pin an input; setting a bit makes it an output.

MCCI Pin Control

PMCPAR Bit	DDRMC Bit	Port MC Signal	MCCI Pin
—	DDM7	PMC7	TXDA
—	DDM6	PMC6	RXDA
—	DDM5	PMC5	TXDB
—	DDM4	PMC4	RXDB
PMCPA3	DDM3	PMC3	$\overline{SS}$
—	DDM2	PMC2	SCK
PMCPA1	DDM1	PMC1	MOSI
PMCPA0	DDM0	PMC0	MISO

7.3 Serial Peripheral Interface

The SPI submodule communicates with external devices through a synchronous serial bus. The SPI is fully compatible with SPI systems found on other Motorola products, but has enhanced capabilities. The SPI can perform full-duplex three-wire or half-duplex two-wire transfers.

7.3.1 SPI Pins

The SPI uses four bidirectional pins. These pins can be configured for general-purpose I/O when not needed for SPI application. When used for SPI functions, the pins should have pull-up resistors. The following table shows SPI pin functions

SPI Pin Function

Pin Names	Mode	Function
Master In Slave Out (MISO)	Master Slave	Provides serial input to the SPI Provides serial output from the SPI
Master Out Slave In (MOSI)	Master Slave	Provides serial output from the SPI Provides serial input to the SPI
Serial Clock (SCK)	Master Slave	Provides clock output from SPI Provides clock input to SPI
Slave Select ($\overline{SS}$)	Master Slave	Causes mode fault Initiates serial transfer

7.3.2 SPI Registers

The programmer's model for the SPI consists of the MCCI global and pin control registers, the SPI control register (SPCR), the SPI status register (SPSR), and the SPI data register (SPDR). All SPI registers can be read and written by the CPU. SPCR must be initialized before the SPI is enabled to ensure defined operation. The SPI is enabled by setting the SPE bit in SPCR. Reset values are shown below each register.

SPCR — SPI Control Register **$YFFC38**

15	14	13	12	11	10	9	8	7							0
SPIE	SPE	WOMP	MSTR	CPOL	CPHA	LSBF	SIZE	SPBR							
RESET:															
0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0

SPCR contains parameters for configuring the SPI. The CPU has read and write access to all control bits, but the MCCI has read access only to all bits except SPE. Writing a new value to SPCR while the SPI is enabled disrupts operation. Writing the same value into SPCR while the SPI is enabled has no effect on SPI operation.

SPIE — SPI Interrupt Enable

0 = SPI interrupts disabled
1 = SPI interrupts enabled

SPE — SPI Enable

0 = SPI is disabled. SPI pins can be used for general-purpose I/O.
1 = SPI is enabled. Pins allocated by PMCPAR are controlled by the SPI.

WOMP — Wired-OR Mode for SPI Pins

0 = Outputs have normal MOS drivers.

1 = Pins designated for output by DDRMC have open-drain drivers.

WOMP allows SPI pins to be connected for wired-OR operation, regardless of whether they are used for general-purpose output or for SPI output. WOMP affects the pins whether the SPI is enabled or disabled.

MSTR — Master/Slave Mode Select

0 = SPI is a slave device and only responds to externally generated serial data.

1 = SPI is system master and can initiate transmission to external SPI devices.

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

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 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.

LSBF — Least Significant Bit First

0 = Serial data transfer starts with MSB

1 = Serial data transfer starts with LSB

SIZE — Transfer Data Size

0 = 8-bit data transfer

1 = 16-bit data transfer

SPBR — SPI Baud Rate

The SPI 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. Giving SPBR a value of zero or one disables the baud rate generator. Use the following expressions to determine baud rate:

$$\text{SCK Baud Rate} = \frac{\text{System Clock Frequency}}{2\,(\text{SPBR Value})}$$

or

$$\text{SPBR Value} = \frac{\text{System Clock Frequency}}{2\,(\text{SCK Baud Rate})}$$

SPSR — SPI Status Register **$YFFC3C**

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

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

SPIF — SPI Finished Flag
- 0 = SPI not finished
- 1 = SPI finished

WCOL — Write Collision
- 0 = No write collision occurred
- 1 = Write collision occurred

MODF — Mode Fault Flag
- 0 = Normal operation
- 1 = Another SPI node requested to become the network SPI master while the SPI was enabled in master mode ($\overline{SS}$ input taken low).

SPDR — SPI Data Register **$YFFC3E**

15							8	7							0
UPPB								LOWB							
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

A write to SPDR initiates transmission or reception in the master device. At the completion of transmission, the SPIF status bit is set in both master and slave devices. Received data is buffered. SPIF must be cleared before a subsequent transfer of data from the shift register to the buffer or overrun occurs. The byte or word that causes overrun is lost. Transmitted data is not buffered. A write to SPDR places data directly into the shift register for transmission.

UPPB — Upper Byte

In 16-bit transfer mode, UPPB is used to access the most significant 8 bits of the data. Bit 15 of the SPDR is the MSB of the 16-bit data.

LOWB — Lower Byte

In 8-bit transfer mode, data is accessed at the address of LOWB. MSB in 8-bit transfer mode is bit 7 of the SPDR. In 16-bit transfer mode, LOWB holds the least significant 8 bits of the data.

7.3.3 SPI Operation

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

In master mode, transmission parameters are set by writing to SPCR; the SPI is enabled by setting SPE; then operation is initiated by writing data to SPDR. In slave mode, operation proceeds in response to $\overline{SS}$ signal assertion by an external bus master. Slave operation is similar to that of master mode.

Normally, the SPI bus performs synchronous bidirectional transfers. The serial clock on the SPI bus master supplies the clock signal (SCK) to time the transfer of data. Four possible combinations of clock phase and polarity can be specified by means of the CPHA and CPOL bits in SPCR. Data can be transferred either LSB or MSB first, depending on the value of the LSBF bit in SPCR. The number of bits transferred per command defaults to eight, but can be set to 16 bits by setting the field in SPCR.

When the SPI finishes a transmission it sets the SPIF flag, clears SPE and stops. If the SPIE bit in SPCR is set, an interrupt request is generated when SPIF is set.

Although the SPI 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.

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

7.4 Serial Communication Interface

There are two identical independent SCI systems in the MCCI, SCIA and SCIB. Each SCI system is a full-duplex universal asynchronous receiver transmitter (UART). Each SCI system is fully compatible with the SCI systems found on other Motorola devices, such as the M68HC11 and M68HC05 Families. The following discussions apply to both SCIA and SCIB. Differences in register addresses and pin names are noted.

7.4.1 SCI Pins

A unidirectional transmit data pin (either TXDA or TXDB) and a unidirectional receive data pin (either RXDA or RXDB) is associated with each SCI. Each pin can be used by the associated SCI or for general-purpose I/O.

SCI pins and their functions are identified in the following table.

Pin Names	Mnemonics	Mode	Function
Receive Data A and B	RXDA, RXDB	Receiver Disabled Receiver Enabled	General-Purpose I/O Serial Data Input to SCI
Transmit Data A and B	TXDA, TXDB	Transmitter Disabled Transmitter Enabled	General-Purpose I/O Serial Data Output from SCI

7.4.2 SCI Registers

The SCI programming model includes the MCCI global and pin control registers, and eight SCI registers. Each SCI contains four registers: two control registers, one status register, and one data register.

All registers can be read or written at any time by the CPU. Rewriting the same value to any SCI register does not disrupt operation; however, writing a different value into an SCI register when the SCI is running may disrupt operation. To change register values, the receiver and transmitter should be disabled with the transmitter allowed to finish first. The status flags in register SCSR can be cleared at any time.

SCCR0A, SCCR0B — SCI Control Register 0 **$YFFC18, $YFFC28**

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

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

SCBR — Baud Rate

SCI baud rate is programmed by writing a 13-bit value to this field. Writing a value of zero to SCBR disables the baud rate generator.

The SCI receiver operates asynchronously. An internal clock is necessary to synchronize the receiver 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:

$$\text{SCI Baud Rate} = \frac{\text{System Clock Frequency}}{32\,(\text{SCBR Value})}$$

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

SCCR1A, SCCR1B — SCI Control Register 1 **$YFFC1A, $YFFC2A**

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

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

SCCR1A/B15 — Not Implemented

LOOPS — Loop Mode

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

1 = Test SCI operation, looping, feedback path enabled

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

ILIE — Idle-Line Interrupt Enable
0 = SCI IDLE interrupts inhibited
1 = SCI IDLE interrupts enabled

TE — Transmitter Enable
0 = SCI transmitter disabled (TXD pin can be used for general-purpose 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 in progress when TE is cleared.

RE — Receiver Enable
0 = SCI receiver disabled (status bits inhibited, RXD pin can be used for general-purpose I/O)
1 = SCI receiver enabled (RXD pin dedicated to SCI)

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.

SCSRA, SCSRB — SCI Status Register **$YFFC1C, $YFFC2C**

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

Each 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, and SCDR must be written or read before the status bit is cleared.

Reading either byte of SCSR causes all 16 bits to be accessed. Any status bit already set in either byte is 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 overwrites 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 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 all three samples are not 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 when 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.

SCDRA, SCDRB — SCI Data Register **$YFFC1E, $YFFC2E**

15 … 9	8	7	6	5	4	3	2	1	0
NOT USED	R8/T8	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0
RESET:	U	U	U	U	U	U	U	U	U

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

8 Standby RAM and TPU Emulation RAM

The two 2-Kbyte RAM modules in the MCU are structurally similar, but functionally different. The TPURAM module has no external standby voltage (V_{STBY}) connection or power-loss flag (PDS), but supports the use of custom TPU microcode. The STBRAM module, on the other hand, has a V_{STBY} connection and provides a power-loss flag and automatic switching to standby power when V_{DD} drops below a specified level, but does not support TPU microcode emulation.

8.1 Overview

Both RAM modules consist of a control register block that is located at a fixed range of addresses in MCU address space, and a 2-Kbyte array of fast (two bus cycle) static RAM that can be mapped to any 2-Kbyte boundary in address space. TPURAM control registers are located at addresses $YFFD00–YFFD3F, while STBRAM module control registers are located from $YFFB00–YFFB3F, as shown in the address map. STBRAM control registers occupy the same addresses as MC68HC16Y1 TPURAM control registers. MC68HC916Y1 TPURAM control register addresses are located in different locations from the MC68HC16Y1 TPURAM control registers. Refer to **1.5 Using the MC68HC916Y1 to Emulate the MC68HC16Y1** for more information.

Both modules respond to program and data space accesses. Data can be read or written in bytes, words, or long words. Arrays must not be mapped so that array addresses overlap module control register addresses, as overlap makes the registers inaccessible. When microcode emulation is unnecessary, the TPURAM array can be mapped to form a contiguous extension of the STBRAM array. While it is possible to map STBRAM over TPURAM while TPURAM is used for microcode emulation, this is not recommended, as this effectively makes a wired-AND connection between the module data bus lines, and can affect accesses to STBRAM.

8.2 RAM Register Blocks

RAM control registers occupy a 64-byte block. There are three control registers in the block: the RAM module configuration register (STBRAMMCR, TRAMMCR), the RAM test register (STBRAMTST, TRAMTST), and the RAM array base address register (STBRAMBAR, TRAMBAR). The rest of the register block contains unimplemented register locations. Unimplemented register addresses are read as zeros, and writes to them have no effect.

STBRAM Control Register Address Map

Address	15 8	7 0
$YFFB00	STBRAM MODULE CONFIGURATION REGISTER (STBRAMMCR)	
$YFFB02	STBRAM TEST REGISTER (STBRAMTST)	
$YFFB04	STBRAM BASE ADDRESS REGISTER (STBRAMBAR)	
$YFFB06–$YFFB3F	NOT IMPLEMENTED	

TPURAM Control Register Address Map

Address	15 8	7 0
$YFFD00	TPURAM MODULE CONFIGURATION REGISTER (TRAMMCR)	
$YFFD02	TPURAM TEST REGISTER (TRAMTST)	
$YFFD04	TPURAM BASE ADDRESS REGISTER (TRAMBAR)	
$YFFD06–$YFFD3F	NOT IMPLEMENTED	

Y = M111, where M is the logic state of the modmap (MM) bit in the SCIMCR.

8.3 RAM Registers

Both STBRAM and TPURAM registers are described in the following paragraphs. Differences are shown by shaded blocks in the diagrams, and are discussed under the appropriate mnemonic.

STBRAMMCR, TRAMMCR — RAM Module Configuration Register **$YFFB00, $YFFD00**

15	14	13	12	11	9	8	7	0
STOP	NOT USED		PDS	NOT USED		RASP	NOT USED	
RESET:								
0			U			U		

Bits in the module configuration register determine whether a RAM module is in low-power stop mode or normal mode, indicate failure of standby RAM power, and determine in which address space the array resides. Reads of unimplemented bits always return zeros. Writes do not affect unimplemented bits.

STOP — Stop Control Bit

0 = RAM array operates normally.

1 = RAM array enters low-power stop mode.

This bit controls whether the RAM array is in low-power consumption mode or operating normally. Reset state is zero, for normal operation. In stop mode, the array retains its contents, but cannot be read or written by the CPU.

PDS — Standby Power Status Bit

0 = Loss of standby power

1 = No loss of standby power

The STBRAM array can be powered by a standby power source, V_{STBY}, while V_{DD} to the microcontroller is turned off. For STBRAMMCR only, PDS indicates when V_{STBY} has fallen below a specified level for a specified period of time. To detect power loss, software must first set PDS, then monitor its state during normal operation and following reset. PDS is not implemented in TRAMMCR, and always reads zero.

RASP — RAM Array Space Field

0 = RAM array is placed in unrestricted space

1 = RAM array is placed in supervisor space.

This bit limits access to the RAM array in microcontrollers that support separate user and supervisor operating modes. Because the CPU16 operates in supervisor mode only, RASP has no effect.

STBRAMTST, TRAMTST — RAM Test Register **$YFFB02, $YFFD02**

Test registers are used for factory test of the RAM modules.

STBRAMBAR, TRAMBAR — RAM Base Address and Status Register **$YFFB04, $YFFD04**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	NOT USED		RAMDS
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0			1

STBRAMBAR and TRAMBAR specify an array base address in the system memory map. STBRAMBAR and TRAMBAR can be written only once after reset, which prevents accidental remapping of the array.

STBRAMBAR[15:3], TRAMBAR[15:3] — RAM Array Base Address Field

This field specifies bits [23:11] of the array base address. To be accessed, the array must be enabled. Because ADDR[23:20] are driven to the same logic state as ADDR19, addresses in the range $080000 to $F7FFFF cannot be accessed by the CPU16. If a RAM array is mapped to these addresses, the system must be reset before the array can be accessed.

RAMDS — RAM Array Disable Status Bit

0 = RAM array is enabled

1 = RAM array is disabled

RAMDS indicates whether an array is active or disabled. The array is disabled after reset. Writing a valid base address into STBRAMBAR or TRAMBAR automatically clears the corresponding RAMDS and enables the array.

8.4 RAM Operation

There are six RAM operating modes, as follows:

A 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.

Standby mode is intended to preserve RAM contents when V_{DD} is removed. In standby mode, STBRAM contents are maintained by V_{STBY}. Circuitry within the STBRAM module detects the change in V_{DD} and switches to V_{STBY} with no loss of data. While STBRAM is powered by V_{STBY}, access to the array is not guaranteed. TPURAM does not have standby power switching. When V_{DD} is removed, TPURAM content is lost.

Reset mode allows the CPU to complete a bus cycle before resetting. When a synchronous reset occurs while a byte or word RAM access is in progress, the access is completed. When reset occurs during the first word access of a long-word operation, only the first word access is completed. When 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 can be corrupted by asynchronous reset.

The test mode functions in conjunction with the SCIM test functions. Test mode is used during factory test of the MCU.

Setting the STOP bit in the appropriate RAMMCR switches a RAM module to low-power mode. In low-power mode, the RAM array retains its contents, but cannot be read or written by the CPU. Because the CPU16 always operates in supervisor mode, STOP can be read or written at any time. STOP is set during reset. Stop mode is exited by clearing STOP. The RAM modules will switch to standby mode while in low-power mode, provided operating constraints discussed above are met.

The TPURAM array can emulate the microcode ROM in the TPU module. This provides a means of developing custom TPU code. The TPU selects TPU emulation mode. While in TPU emulation mode, the access timing of the TPURAM module matches the timing of the TPU microinstruction ROM to ensure accurate emulation. Normal accesses through the IMB are inhibited and the control registers have no effect, allowing external RAM to emulate the TPURAM at the same addresses. Refer to **4 Time Processor Unit** for more information. The STBRAM module cannot be used for TPU microcode emulation.

9 Flash EEPROM

The 48-Kbyte flash electrically-erasable programmable read-only memory module (FLASH) 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 Flash EEPROM module consists of a control-register block that occupies a fixed position in MCU address space, and a 48-Kbyte EPROM array that can be placed in the lower 48 Kbytes of any 64-Kbyte block in 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 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. Hardware interlocks protect stored data from corruption if the program/erase voltage to the flash EEPROM array is enabled accidently.

Flash EEPROM Address Map

Address	15 8	7 0
$YFF820	FLASH EEPROM MODULE CONFIGURATION REGISTER (FEEMCR)	
$YFF822	FLASH EEPROM TEST REGISTER (FEETST)	
$YFF824	FLASH EEPROM BASE ADDRESS HIGH REGISTER (FEEBAH)	
$YFF826	FLASH EEPROM BASE ADDRESS LOW REGISTER (FEEBAL)	
$YFF828	FLASH EEPROM CONTROL REGISTER (FEECTL)	
$YFF82A	RESERVED	
$YFF82C	RESERVED	
$YFF82E	RESERVED	
$YFF830	FLASH EEPROM BOOTSTRAP WORD 0 (FEEBS0)	
$YFF832	FLASH EEPROM BOOTSTRAP WORD 1 (FEEBS1)	
$YFF834	FLASH EEPROM BOOTSTRAP WORD 2 (FEEBS2)	
$YFF836	FLASH EEPROM BOOTSTRAP WORD 3 (FEEBS3)	
$YFF838	RESERVED	
$YFF83A	RESERVED	
$YFF83C	RESERVED	
$YFF83E	RESERVED	

Y = M111, where M is the logic state of the modmap (MM) bit in the SCIMCR

Flash EEPROM module control registers in the MC68HC916Y1 occupy the same locations as masked ROM module control registers in the MC68HC16Y1, and the FLASH module can be used to emulate the MRM. However, the FLASH module does not provide the $\overline{\text{CSM}}$ memory emulation mode chip-select line. For more information refer to **1.5 Using the MC68HC916Y1 to Emulate the MC68HC16Y1**. Holding pin DATA14 low during reset disables the 48-Kbyte flash EEPROM module and places it in stop mode.

9.2 Flash EEPROM Control Block

The flash EEPROM control block contains five registers: a module configuration register (FEEMCR), a test register (FEETST), two array base address registers (FEEBAH and FEEBAL), and a control register (FEECTL). Four additional words in the control block can contain bootstrap information when the flash EEPROM is used as bootstrap memory.

Each register in the control block has an associated shadow register that is physically located in a spare flash EEPROM 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 a location in the flash EEPROM 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 flash EEPROM array is erased.

Configuration information is specified and programmed independently of the flash EEPROM 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 can be written only when the LOCK bit in the FEEMCR is disabled or when the STOP bit in the FEEMCR is set. These restrictions are noted in the individual register descriptions.

9.3 Flash EEPROM Array

The base address registers specify the base address of the flash EEPROM array. A default reset base address can be programmed into the base address shadow register. The array base address must be on a 64-Kbyte boundary. Because ADDR[23:20] are driven to the same logic state as ADDR19, addresses in the range $080000 to $F7FFFF cannot be accessed by the CPU16. If the flash EEPROM 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. Should a portion of the array overlap the flash EEPROM register block, the registers remain accessible, but accesses to that portion of the array are ignored. However, should the array overlap the control block of another module, those registers may become inaccessible.

9.4 Flash EEPROM Registers

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

FEEMCR — Flash EEPROM Module Configuration Register **$YFF820**

15	14	13	12	11	10	9	8	7	6	5	0
STOP	FRZ	0	BOOT	LOCK	0	ASPC		WAIT		NOT USED	
RESET:											
SB	0	0	SB	SB	0	SB		SB			

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, program/erase voltage is automatically turned off. However, the enable programming/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

BOOT — Boot Control

0 = Flash EEPROM module control words respond to bootstrap vector addresses

1 = Flash EEPROM module control words do not respond to bootstrap vector addresses

On reset, the BOOT bit takes on the default value stored in the shadow register. If BOOT = 0 and STOP = 0, the module responds to program space accesses of IMB addresses $000000 to $000006 following reset, and the contents of FEEBS[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 — Flash EEPROM 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. In systems with restricted access levels, ASPC also determines whether accesses are restricted to supervisor space. The field can be written 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 flash EEPROM array to supervisor or user space, and to program or data space.

WAIT — Wait States

The WAIT field specifies the number of wait states inserted during accesses to the flash EEPROM module. A wait state has the duration of one system clock cycle. This field affects both control block and array access.

WAIT[1:0]	Wait States	Clocks/Transfer
00	1	3
01	2	4
10	3	5
11	0	2

The value of the WAIT field is compatible with the lower two bits of the DSACK field in the SCIM chip select option registers.

FEETST — Flash EEPROM Test Register **$YFF822**

This register is used for factory test purposes only.

FEEBAH — Flash EEPROM Base Address High Register **$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:	SHADOW BIT DEFAULT VALUE							

FEEBAH contains the 16 high-order bits of the flash EEPROM array base address. During reset, FEEBAH takes on the default value programmed into the associated shadow register. After reset, if LOCK = 0 and STOP = 1, software can write to FEEBAH and FEEBAL to relocate the flash EEPROM array. Because ADDR[23:20] are driven to the same logic state as ADDR19, addresses in the range $080000 to $F7FFFF cannot be accessed by the CPU16. If the flash EEPROM array is mapped to these addresses, the system must be reset before the array can be accessed.

FEEBAL — Flash EEPROM Base Address Low Register **$YFF826**

15 … 0
NOT USED

Because the flash EEPROM array in the MC68HC916Y1 is mapped to 64-Kbyte boundaries, FEEBAL is not used. It cannot be written, and always reads all zeros.

FEECTL — Flash EEPROM Control Register **$YFF828**

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

FEECTL contains the bits needed to control the programming and erasure of the flash EEPROM. This register is accessible in supervisor mode only.

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 flash EEPROM 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 flash EEPROM locations occur.

ERAS — Erase Control

0 = Flash EEPROM configured for programming

1 = Flash EEPROM configured for erasure

Asserting ERAS causes all locations in the array and all flash EEPROM control 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 flash EEPROM address and data buses are connected to the IMB address and data buses and the flash EEPROM is configured for normal reads. When LAT is set, the flash EEPROM address and data buses are connected to parallel internal latches and the flash EEPROM array is configured for programming or erasing.

Once LAT is set, the next write to a valid flash EEPROM module 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).

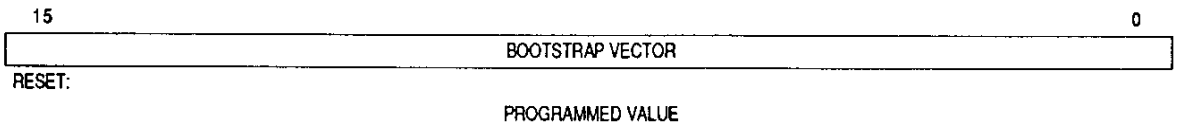
ENPE — Enable Programming/Erase

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 flash EEPROM array location in the flash EEPROM module are ignored.

FEEBS[3:0] — Flash EEPROM Bootstrap Words **$YFF830–$YFF837**



These words can be used as system bootstrap vectors. When the BOOT bit in FEEMCR = 1 during reset, the flash EEPROM module responds to program space accesses of IMB addresses $000000 to $000006 after reset. When BOOT = 0, the flash EEPROM module responds only to normal array and register accesses. FEEBS[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 Flash EEPROM Operation

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

9.5.1 Reset Operation

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

When the state of the STOP shadow bit is zero, the STOP bit in FEEMCR is cleared during reset, and the module responds to accesses in the range specified by FEEBAH. 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 FEEMCR is set during reset and the flash EEPROM 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 flash EEPROM 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 BOOT and STOP bits in FEEMCR are cleared during reset, the flash EEPROM module is configured to respond to bootstrap vector accesses. Vector assignments are as follows:

EEPROM Bootstrap Word	IMB Vector Address	MCU Reset Vector Content
FEEBS0	$000000	Initial ZK, SK, and PK
FEEBS1	$000002	Initial PC
FEEBS2	$000004	Reset — Initial SP
FEEBS3	$000006	Initial IZ

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

9.5.3 Normal Operation

The flash EEPROM 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 FEEMCR 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 FEEMCR. When the flash EEPROM module 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 flash 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 flash EEPROM 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 flash EEPROM bits. Programming is restricted to a single byte or aligned word at a time. The entire flash EEPROM 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 VPP pin. $V_{PP} \geq (V_{DD} - 0.3\ V)$ must be applied at all times or damage to the FLASH module can occur.

9.5.4.1 Intelligent Programming and Erasing

Intelligent programming and erasing procedures verify the flash EEPROM 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 flash EEPROM.

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 FEECTL 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 EEPROM 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 flash EEPROM location is programmed, calculate $t_{pmargin}$ and repeat steps 4 through 7. If the flash EEPROM location does not remain programmed, the flash EEPROM 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 flash EEPROM module 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 flash EEPROM 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 flash EEPROM is erased.
9. If all of the flash EEPROM 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 flash EEPROM locations are erased, calculate $t_{emargin}$ and repeat steps 4 through 8. If all flash EEPROM locations do not remain erased, the flash EEPROM module may be bad.
11. Clear LAT, ERAS, and VFPE to allow normal access to the flash EEPROM.
12. Turn off V_{fp} (reduce voltage on V_{FPE48K} pin to V_{DD}).

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

◊ 1ATX31312-0 PRINTED IN USA 4/93 IMPERIAL LITHO 91079 18,000 MCU YGACAA

MC68HC916Y1TS/D