Order this document
by MC68HC16Y1TS/D Rev. 1

MC68HC16Y1

Technical Summary

16-Bit Modular Microcontroller

1 Introduction

The MC68HC16Y1 is a high-speed 16-bit control unit 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 internal bus. Standardization facilitates rapid development of devices tailored for specific applications.

The MC68HC16Y1 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), and a 48 Kbyte masked ROM module (MRM). These modules are interconnected by the Motorola intermodule bus (IMB).

The MC68HC16Y1 can either synthesize an internal clock signal from an external reference, or use an external clock input directly. Operation with a 32.768 kHz reference frequency is standard, but operation with a 4.0 MHz reference is available as an option — contact your Motorola representative for more information. System hardware and software allow changes in clock rate during operation. Because the MC68HC16Y1 is a fully static design, register and memory contents are not affected by clock rate changes.

High-density complementary metal-oxide semiconductor (HCMOS) architecture makes the basic power consumption of the MC68HC16Y1 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.

Table 1 Ordering Information

Package Type	Frequency (MHz)	Temperature	Order Number
Plastic Surface Mount FC Suffix	16.78	–40° to +85°C	M68HC16Y1CFC

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

TABLE OF CONTENTS

1.1 Features

- 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
  - Single-Chip or Expanded Modes of Operation
  - External Bus Support in Expanded Mode
  - Nine Programmable Chip Select Outputs
  - System Protection Logic
  - Watchdog Timer, Clock Monitor, and Bus Monitor
  - Parallel Ports Option On Address and Data Bus in Single-Chip Mode
  - PLL Clock System
- Time Processor Unit
  - 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
  - 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
  - Eight Channels, Eight Result Registers
  - Eight Automated Modes
  - Three Result Alignment Modes
- Multichannel Communication Interface
  - Dual Serial Communication Interface
  - Serial Peripheral Interface
- TPU Emulation RAM
  - 2 Kbyte Static RAM
  - External Standby Voltage Supply Input
- Masked ROM
  - 48 Kbyte 16-Bit Array
  - User-Selectable Default Base Address
  - User-Selectable Bootstrap ROM Function
  - User-Selectable ROM Verification Code

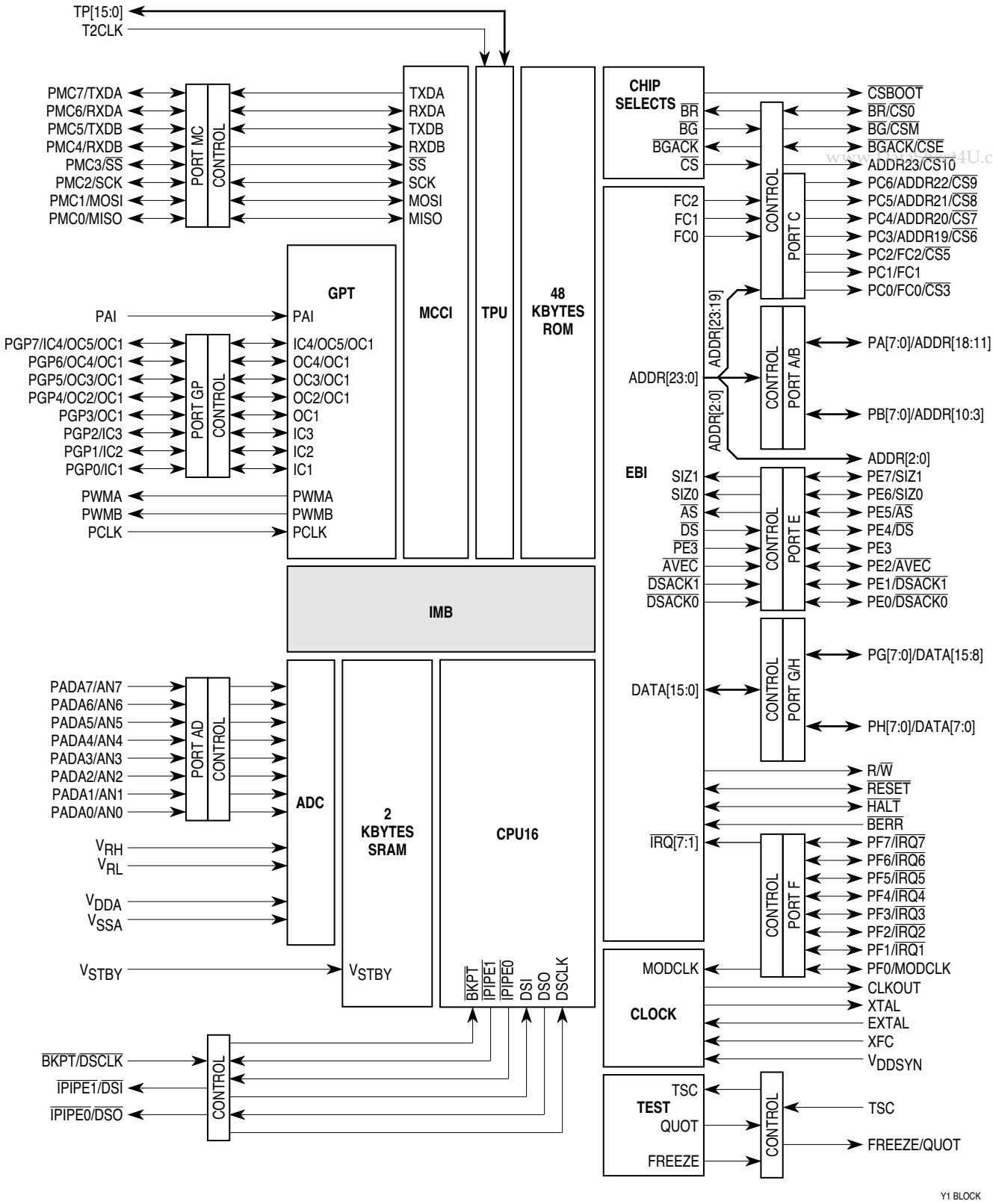


Figure 1 MC68HC16Y1 Block Diagram

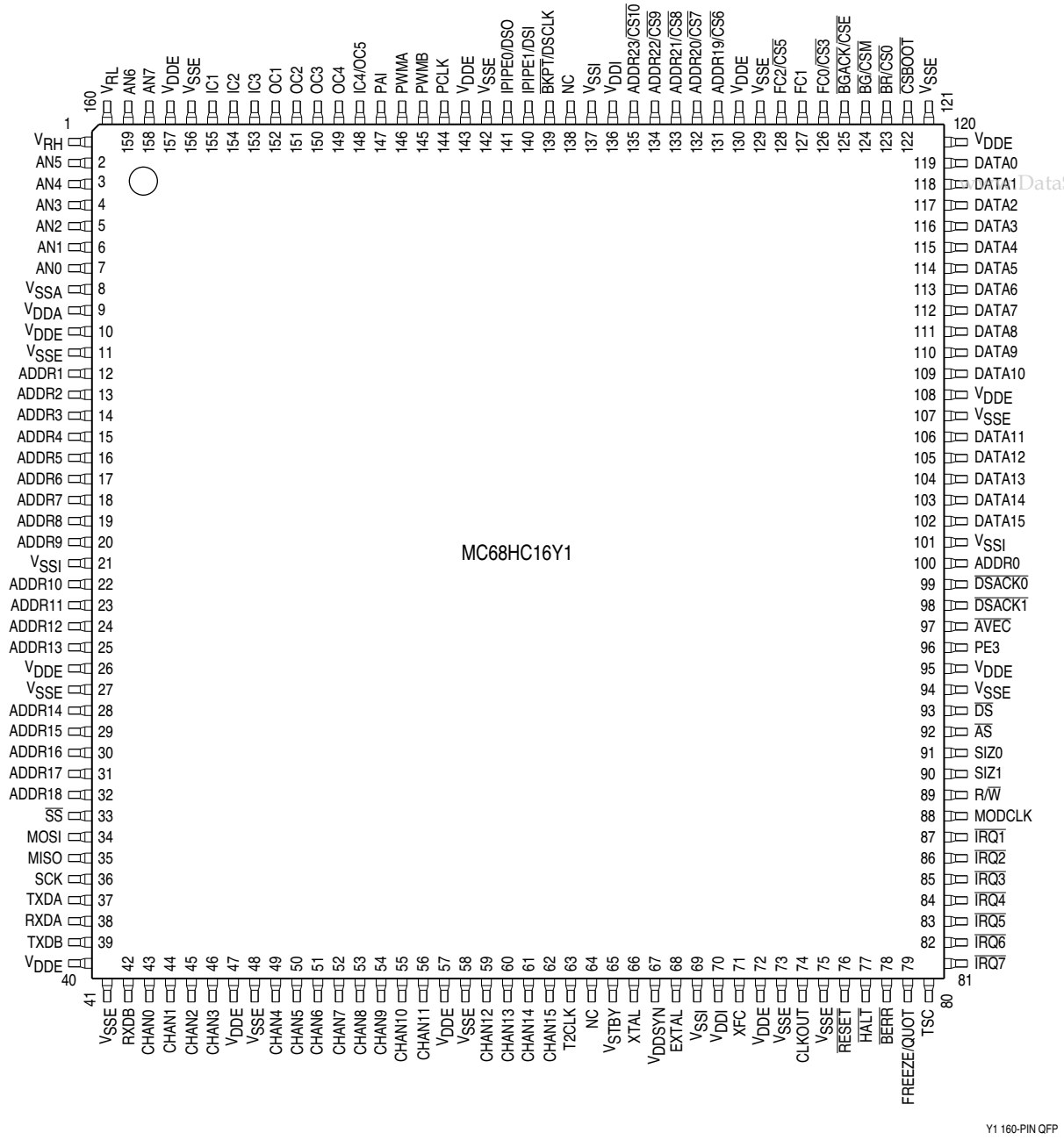


Figure 2 MC68HC16Y1 160-Pin QFP Pinout

1.2 Pin Description

The table below describes MC68HC16Y1 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 **Table 3** for a description of output drivers. An entry in the Discrete I/O column of **Table 2** indicates that a pin has an alternate I/O function — port designation is given when it applies. Refer to **Figure 1** for port organization.

Table 2 MC68HC16Y1 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	—	—
$\overline{BG/CSM}$	B	—	—	—	—
$\overline{BGACK/CSE}$	B	Y	N	—	—
$\overline{BKPT}$/DSCLK	—	Y	Y	—	—
$\overline{BR/CS0}$	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]
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	—
$\overline{SS}$	Bo	Y	Y	I/O	MC3
PE3	B	Y	Y	I/O	E3

Table 2 MC68HC16Y1 Pin Characteristics (Continued)

Pin Mnemonic	Output Driver	Input Synchronized	Input Hysteresis	Discrete I/O	Port Designation
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]
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.

Table 3 MC68HC16Y1 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.

1. Pins with this type of driver may 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.

Table 4 MC68HC16Y1 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

1.3 Address Map

The internal address map of the MC68HC16Y1 is shown below. Although there are 24 intermodule bus (IMB) address lines, the CPU16 uses only ADDR[19:0]. ADDR[23:20] follow the logic state of 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.

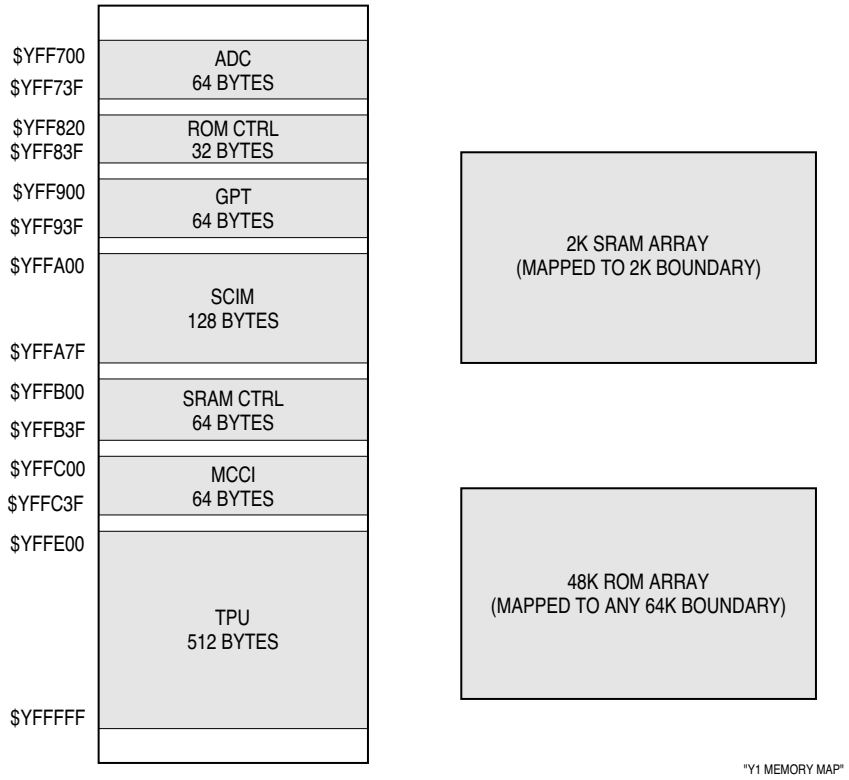


Figure 3 MC68HC16Y1 Address Map

In the address map, Y = M111, where M is the modmap signal state on the IMB. M reflects the state of the modmap bit in the module configuration register of the single-chip integration module. In the MC68HC16Y1, Y must equal $F — if M is cleared, IMB modules will be inaccessible until a reset occurs. M 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 to support exception processing, address space partitioning, multiple interrupt levels, and vectored interrupts. The standardized modules in the MC68HC16Y1 communicate with one another and with external components via the IMB. Although the full IMB supports 24 address and 16 data lines, the MC68HC16Y1 uses only 16 data lines and 20 address lines. Because the CPU16 uses only 20 address lines, ADDR[23:20] are tied to ADDR19 when processor driven. ADDR[23:20] are brought out to pins for test purposes.

2 CPU16

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

2.1 Overview

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

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

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

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

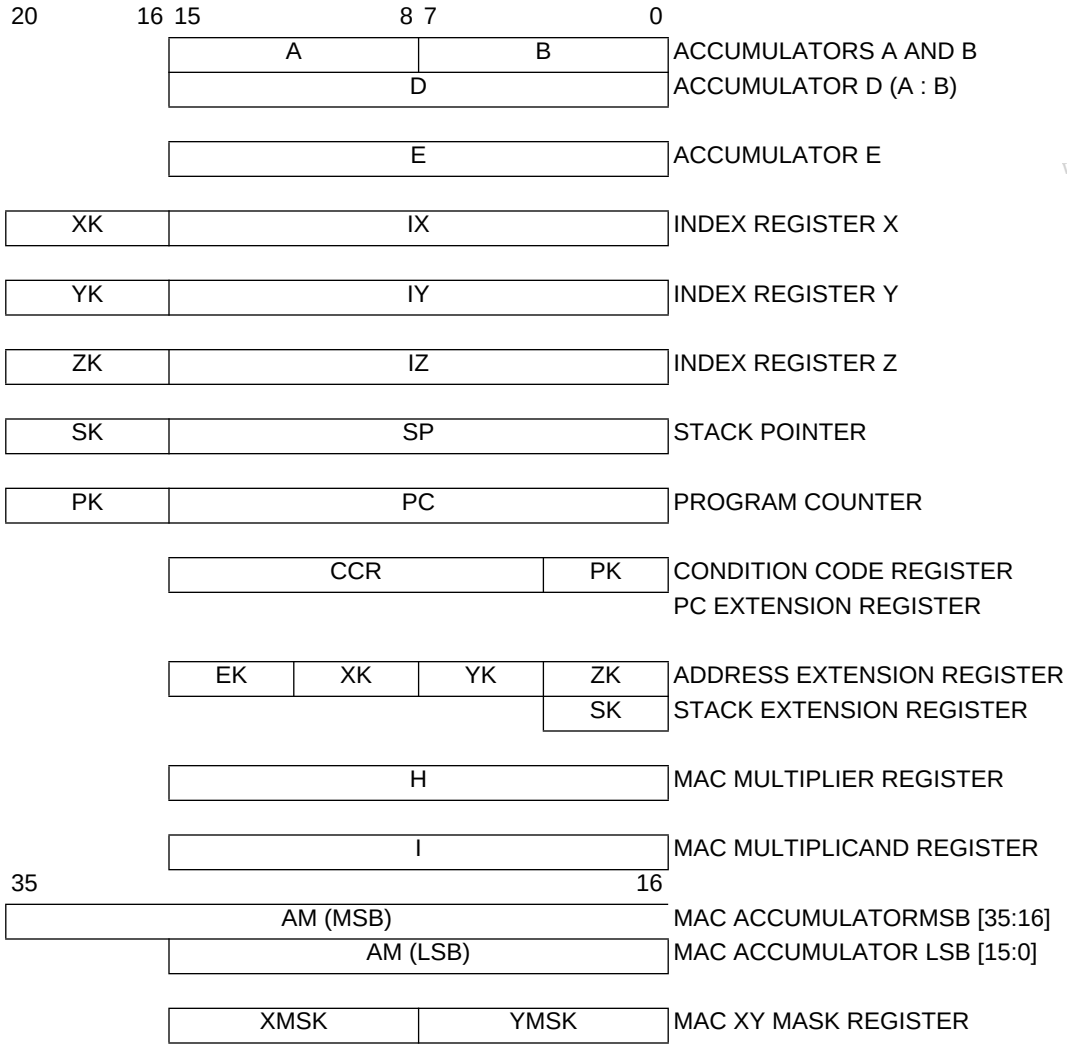
2.2 M68HC11 Compatibility

CPU16 architecture is a superset of M68HC11 architecture. All M68HC11 resources are available in the HC16. M68HC11 instructions are either directly implemented in the M68HC16, or have been replaced by instructions with an equivalent form — the instruction sets are source code compatible. Some instructions are executed differently in the M68HC16. 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

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

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

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

MV — Accumulator M overflow flag
Set when overflow into the accumulator M sign bit (AM35) has occurred.

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

EV — Extension Bit Overflow Flag
Set when an overflow into bit 31 of accumulator M has occurred.

N — Negative Flag
Set when the MSB of a result register is set.

Z — Zero Flag
Set when all bits of a result register are zero.

V — Overflow Flag
Set when twos 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 may be accessed on odd byte boundaries, with a substantial performance penalty.

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

2.6 Addressing Modes

The CPU16 provides 10 types of addressing. Each type encompasses one or more addressing modes. Six CPU16 addressing types are identical to M68HC11 addressing types.

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

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

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

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

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

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

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

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

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

Relative modes are used for branch and long branch instructions. A byte or word signed twos 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 will have a significantly smaller instruction space.

The instruction set is based upon that of the M68HC11, but the opcode map has been rearranged to maximize performance with a 16-bit data bus. All M68HC11 instructions are supported by the CPU16, although they may be executed differently. Most M68HC11 code will run on the CPU16 following reassembly. The user must take into account changed instruction times, the interrupt mask, and the new interrupt stack frame.

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

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

Table 5 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								

Table 5 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								

Table 5 Instruction Set Summary (Continued)

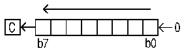
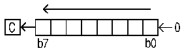
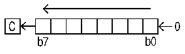
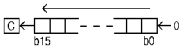
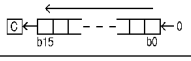
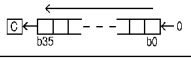
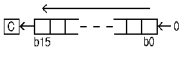
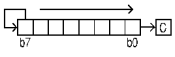
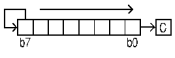
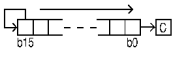
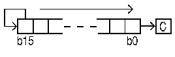
Mnemonic	Operation	Description	Address Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
ANDD	AND D	(D) • (M : M + 1) ⇒ D	IND8, X	86	ff	6	—	—	—	—	Δ	Δ	0	—
			IND8, Y	96	ff	6								
			IND8, Z	A6	ff	6								
			E, X	2786	—	6								
			E, Y	2796	—	6								
			E, Z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

Table 5 Instruction Set Summary (Continued)

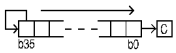
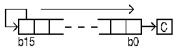
Mnemonic	Operation	Description	Address Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
ASRM	Arithmetic Shift Right AM		INH	27BA	—	4	—	—	—	Δ	Δ	—	—	Δ
ASRW	Arithmetic Shift Right Word		IND16, X IND16, Y IND16, Z EXT	270D 271D 272D 273D	gggg gggg gggg hh ll	8 8 8 8	—	—	—	—	Δ	Δ	Δ	Δ
BCC[4]	Branch if Carry Clear	If C = 0, branch	REL8	B4	rr	6, 2	—	—	—	—	—	—	—	—
BCLR	Clear Bit(s)	(M) • ($\overline{\text{Mask}}$) ⇒ M	IND16, X IND16, Y IND16, Z EXT IND8, X IND8, Y IND8, Z	08 18 28 38 1708 1718 1728	mm gggg mm gggg mm gggg mm hh ll mm ff mm ff mm ff	8 8 8 8 8 8 8	—	—	—	—	Δ	Δ	0	—
BCLRW	Clear Bit(s) Word	(M : M + 1) • ($\overline{\text{Mask}}$) ⇒ M : M + 1	IND16, X IND16, Y IND16, Z EXT	2708 2718 2728 2738	gggg mmmm gggg mmmm gggg mmmm hh ll mmmm	10 10 10 10	—	—	—	—	Δ	Δ	0	—
BCS[4]	Branch if Carry Set	If C = 1, branch	REL8	B5	rr	6, 2	—	—	—	—	—	—	—	—
BEQ[4]	Branch if Equal	If Z = 1, branch	REL8	B7	rr	6, 2	—	—	—	—	—	—	—	—
BGE[4]	Branch if Greater Than or Equal to Zero	If N ⊕ V = 0, branch	REL8	BC	rr	6, 2	—	—	—	—	—	—	—	—
BGND	Enter Background Debug Mode	If BDM enabled enter BDM; else, illegal instruction	INH	37A6	—	—	—	—	—	—	—	—	—	—
BGT [4]	Branch if Greater Than Zero	If Z + (N ⊕ V) = 0, branch	REL8	BE	rr	6, 2	—	—	—	—	—	—	—	—
BHI [4]	Branch if Higher	If C + Z = 0, branch	REL8	B2	rr	6, 2	—	—	—	—	—	—	—	—
BITA	Bit Test A	(A) • (M)	IND8, X IND8, Y IND8, Z IMM8 IND16, X IND16, Y IND16, Z EXT E, X E, Y E, Z	49 59 69 79 1749 1759 1769 1779 2749 2759 2769	ff ff ff ii gggg gggg gggg hh ll — — —	6 6 6 2 6 6 6 6 6 6 6	—	—	—	—	Δ	Δ	0	—
BITB	Bit Test B	(B) • (M)	IND8, X IND8, Y IND8, Z IMM8 IND16, X IND16, Y IND16, Z EXT E, X E, Y E, Z	C9 D9 E9 F9 17C9 17D9 17E9 17F9 27C9 27D9 27E9	ff ff ff ii gggg gggg gggg hh ll — — —	6 6 6 2 6 6 6 6 6 6 6	—	—	—	—	Δ	Δ	0	—
BLE [4]	Branch if Less Than or Equal to Zero	If Z + (N ⊕ V) = 1, branch	REL8	BF	rr	6, 2	—	—	—	—	—	—	—	—
BLS[4]	Branch if Lower or Same	If C + Z = 1, branch	REL8	B3	rr	6, 2	—	—	—	—	—	—	—	—
BLT[4]	Branch if Less Than Zero	If N ⊕ V = 1, branch	REL8	BD	rr	6, 2	—	—	—	—	—	—	—	—
BMI [4]	Branch if Minus	If N = 1, branch	REL8	BB	rr	6, 2	—	—	—	—	—	—	—	—
BNE [4]	Branch if Not Equal	If Z = 0, branch	REL8	B6	rr	6, 2	—	—	—	—	—	—	—	—

Table 5 Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
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 rrrr	10, 14								
			IND16, Y	1A	mm gggg rrrr	10, 14								
			IND16, Z	2A	mm gggg rrrr	10, 14								
			EXT	3A	mm hh ll rrrr	10, 14								
BRN	Branch Never	If 1 = 0, branch	REL8	B1	rr	2	—	—	—	—	—	—	—	—
BRSET[4]	Branch if Bit(s) Set	If (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 rrrr	10, 14								
			IND16, Y	1B	mm gggg rrrr	10, 14								
			IND16, Z	2B	mm gggg rrrr	10, 14								
			EXT	3B	mm hh ll rrrr	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	—	—	—	—

Table 5 Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
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								
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	One's 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	One's Complement A	$FF – (A) ⇒ A	INH	3700	—	2	—	—	—	—	Δ	Δ	0	1
COMB	One's Complement B	$FF – (B) ⇒ B	INH	3710	—	2	—	—	—	—	Δ	Δ	0	1
COMD	One's Complement D	$FFFF – (D) ⇒ D	INH	27F0	—	2	—	—	—	—	Δ	Δ	0	1
COME	One's Complement E	$FFFF – (E) ⇒ E	INH	2770	—	2	—	—	—	—	Δ	Δ	0	1
COMW	One's 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								

Table 5 Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
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								
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								

Table 5 Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
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								
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	03	ff	8	—	—	—	—	Δ	Δ	Δ	—
			IND8, Y	13	ff	8								
			IND8, Z	23	ff	8								
			IND16, X	1703	gggg	8								
			IND16, Y	1713	gggg	8								
			IND16, Z	1723	gggg	8								
			EXT	1733	hh ll	8								
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	2703	gggg	8	—	—	—	—	Δ	Δ	Δ	—
			IND16, Y	2713	gggg	8								
			IND16, Z	2723	gggg	8								
			EXT	2733	hh ll	8								
JMP	Jump	⟨ea⟩ ⇒ PK : PC	IND20, X	4B	zg gggg	8	—	—	—	—	—	—	—	—
			IND20, Y	5B	zg gggg	8								
			IND20, Z	6B	zg gggg	8								
			EXT20	7A	zb hh ll	6								
JSR	Jump to Subroutine	Push (PC) (SK : SP) – 2 ⇒ SK : SP Push (CCR) (SK : SP) – 2 ⇒ SK : SP ⟨ea⟩ ⇒ PK : PC	IND20, X	89	zg gggg	12	—	—	—	—	—	—	—	—
			IND20, Y	99	zg gggg	12								
			IND20, Z	A9	zg gggg	12								
			EXT20	FA	zb hh ll	10								
LBCC[4]	Long Branch if Carry Clear	If C = 0, branch	REL16	3784	rrrr	6, 4	—	—	—	—	—	—	—	—
LBCS[4]	Long Branch if Carry Set	If C = 1, branch	REL16	3785	rrrr	6, 4	—	—	—	—	—	—	—	—
LBEQ[4]	Long Branch if Equal	If Z = 1, branch	REL16	3787	rrrr	6, 4	—	—	—	—	—	—	—	—
LBEV[4]	Long Branch if EV Set	If EV = 1, branch	REL16	3791	rrrr	6, 4	—	—	—	—	—	—	—	—
LBGE[4]	Long Branch if Greater Than or Equal to Zero	If N ⊕ V = 0, branch	REL16	378C	rrrr	6, 4	—	—	—	—	—	—	—	—
LBGT[4]	Long Branch if Greater Than Zero	If Z ✣ (N ⊕ V) = 0, branch	REL16	378E	rrrr	6, 4	—	—	—	—	—	—	—	—

Table 5 Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
LBHI [4]	Long Branch if Higher	If C ✣ Z = 0, branch	REL16	3782	rrrr	6, 4	—	—	—	—	—	—	—	—
LBLE [4]	Long Branch if Less Than or Equal to Zero	If Z ✣ (N ⊕ V) = 1, branch	REL16	378F	rrrr	6, 4	—	—	—	—	—	—	—	—
LBLS [4]	Long Branch if Lower or Same	If C ✣ Z = 1, branch	REL16	3783	rrrr	6, 4	—	—	—	—	—	—	—	—
LBLT [4]	Long Branch if Less Than Zero	If N ⊕ V = 1, branch	REL16	378D	rrrr	6, 4	—	—	—	—	—	—	—	—
LBMI [4]	Long Branch if Minus	If N = 1, branch	REL16	378B	rrrr	6, 4	—	—	—	—	—	—	—	—
LBMV [4]	Long Branch if MV Set	If MV = 1, branch	REL16	3790	rrrr	6, 4	—	—	—	—	—	—	—	—
LBNE [4]	Long Branch if Not Equal	If Z = 0, branch	REL16	3786	rrrr	6, 4	—	—	—	—	—	—	—	—
LBPL [4]	Long Branch if Plus	If N = 0, branch	REL16	378A	rrrr	6, 4	—	—	—	—	—	—	—	—
LBRA	Long Branch Always	If 1 = 1, branch	REL16	3780	rrrr	6	—	—	—	—	—	—	—	—
LBRN	Long Branch Never	If 1 = 0, branch	REL16	3781	rrrr	6	—	—	—	—	—	—	—	—
LBSR	Long Branch to Subroutine	Push (PC) (SK : SP) – 2 ⇒ SK : SP Push (CCR) (SK : SP) – 2 ⇒ SK : SP (PK : PC) + Offset ⇒ PK : PC	REL16	27F9	rrrr	10	—	—	—	—	—	—	—	—
LBVC [4]	Long Branch if Overflow Clear	If V = 0, branch	REL16	3788	rrrr	6, 4	—	—	—	—	—	—	—	—
LBVS [4]	Long Branch if Overflow Set	If V = 1, branch	REL16	3789	rrrr	6, 4	—	—	—	—	—	—	—	—
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	—	—	—	—	—	—	—	—

Table 5 Instruction Set Summary (Continued)

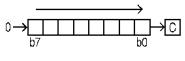
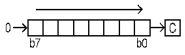
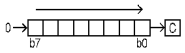
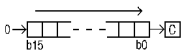
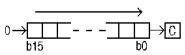
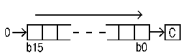
Mnemonic	Operation	Description	Address Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
LDHI	Initialize H and I	$(M : M + 1)_X \Rightarrow H\ R$ $(M : M + 1)_Y \Rightarrow I\ R$	EXT	27B0	—	8	—	—	—	—	—	—	—	—
LDS	Load SP	$(M : M + 1) \Rightarrow SP$	IND8, X	CF	ff	6	—	—	—	—	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	$(M : M + 1) \Rightarrow IX$	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	$(M : M + 1) \Rightarrow IY$	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	$(M : M + 1) \Rightarrow IZ$	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								
LPSTOP	Low Power Stop	If $\overline{S}$ then STOP else NOP	INH	27F1	—	4, 20	—	—	—	—	—	—	—	—
LSR	Logical Shift Right		IND8, X	0F	ff	8	—	—	—	—	0	Δ	Δ	Δ
			IND8, Y	1F	ff	8								
			IND8, Z	2F	ff	8								
			IND16, X	170F	gggg	8								
			IND16, Y	171F	gggg	8								
			IND16, Z	172F	gggg	8								
			EXT	173F	hh ll	8								
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	270F	gggg	8	—	—	—	—	0	Δ	Δ	Δ
			IND16, Y	271F	gggg	8								
			IND16, Z	272F	gggg	8								
			EXT	273F	hh ll	8								

Table 5 Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
MAC	Multiply and Accumulate Signed 16-Bit Fractions	(HR) ∗ (IR) ⇒ E : D (AM) + (E : D) ⇒ AM Qualified (IX) ⇒ IX Qualified (IY) ⇒ IY (HR) ⇒ IZ $(M : M + 1)_X \Rightarrow HR$ $(M : M + 1)_Y \Rightarrow IR$	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	—
ORAB	OR B	(B) ✣ (M) ⇒ B	IND8, X IND8, Y IND8, Z IMM8 IND16, X IND16, Y IND16, Z EXT E, X E, Y E, Z	C7 D7 E7 F7 17C7 17D7 17E7 17F7 27C7 27D7 27E7	ff ff ff ii gggg gggg gggg hh ll — — —	6 6 6 2 6 6 6 6 6 6 6	—	—	—	—	Δ	Δ	0	—
ORD	OR D	(D) ✣ (M : M + 1) ⇒ D	IND8, X IND8, Y IND8, Z E, X E, Y E, Z IMM16 IND16, X IND16, Y IND16, Z EXT	87 97 A7 2787 2797 27A7 37B7 37C7 37D7 37E7 37F7	ff ff ff — — — jj kk gggg gggg gggg hh ll	6 6 6 6 6 6 4 6 6 6 6	—	—	—	—	Δ	Δ	0	—

Table 5 Instruction Set Summary (Continued)

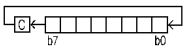
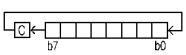
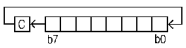
Mnemonic	Operation	Description	Address Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
ORE	OR E	(E) ✣ (M : M + 1) ⇒ E	IMM16 IND16, X IND16, Y IND16, Z EXT	3737 3747 3757 3767 3777	jj kk gggg gggg gggg hh ll	4 6 6 6 6	—	—	—	—	Δ	Δ	0	—
ORP [1]	OR Condition Code Register	(CCR) ✣ IMM16 ⇒ CCR	IMM16	373B	jj kk	4	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
PSHA	Push A	(SK : SP) + 1 ⇒ SK : SP Push (A) (SK : SP) – 2 ⇒ SK : SP	INH	3708	—	4	—	—	—	—	—	—	—	—
PSHB	Push B	(SK : SP) + 1 ⇒ SK : SP Push (B) (SK : SP) – 2 ⇒ SK : SP	INH	3718	—	4	—	—	—	—	—	—	—	—
PSHM	Push Multiple Registers Mask bits: 0 = D 1 = E 2 = IX 3 = IY 4 = IZ 5 = K 6 = CCR 7 = (reserved)	For mask bits 0 to 7: If mask bit set Push register (SK : SP) – 2 ⇒ SK : SP	IMM8	34	ii	4 + 2N N = number of iterations	—	—	—	—	—	—	—	—
PSHMAC	Push MAC State	MAC Registers ⇒ Stack	INH	27B8	—	14	—	—	—	—	—	—	—	—
PULA	Pull A	(SK : SP) + 2 ⇒ SK : SP Pull (A) (SK : SP) – 1 ⇒ SK : SP	INH	3709	—	6	—	—	—	—	—	—	—	—
PULB	Pull B	(SK : SP) + 2 ⇒ SK : SP Pull (B) (SK : SP) – 1 ⇒ SK : SP	INH	3719	—	6	—	—	—	—	—	—	—	—
PULM [1]	Pull Multiple Registers Mask bits: 0 = CCR[15:4] 1 = K 2 = IZ 3 = IY 4 = IX 5 = E 6 = D 7 = (reserved)	For mask bits 0 to 7: If mask bit set (SK : SP) + 2 ⇒ SK : SP Pull register	IMM8	35	ii	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; $(M : M + 1)_X \Rightarrow H$; $(M : M + 1)_Y \Rightarrow I$ (E) – 1 ⇒ E	IMM8	FB	xoyo	6 + 12 per iteration	—	Δ	—	Δ	—	—	—	—
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

Table 5 Instruction Set Summary (Continued)

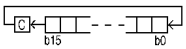
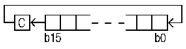
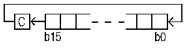
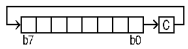
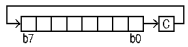
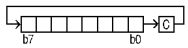
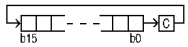
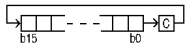
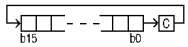
Mnemonic	Operation	Description	Address Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
ROLD	Rotate Left D		INH	27FC	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ROLE	Rotate Left E		INH	277C	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ROLW	Rotate Left Word		IND16, X IND16, Y IND16, Z EXT	270C 271C 272C 273C	gggg 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

Table 5 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	—	—	—	—	—	—	—	—

Table 5 Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
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								
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								

Table 5 Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
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	—	—	—	—	—	—	—	—
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

Table 5 Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
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	—	—	—	—	—	—	—	—
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.

Table 6 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

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

+	—	Addition
-	—	Subtraction or negation (2's complement)
•	—	Multiplication
/	—	Division
>	—	Greater
<	—	Less
=	—	Equal
≥	—	Equal or greater
≤	—	Equal or less
≠	—	Not equal

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

3 Single-Chip Integration Module

The single-chip integration module (SCIM) consists of six submodules that control system start-up, initialization, configuration, and external bus with a minimum of external devices. A block diagram of the SCIM is shown below.

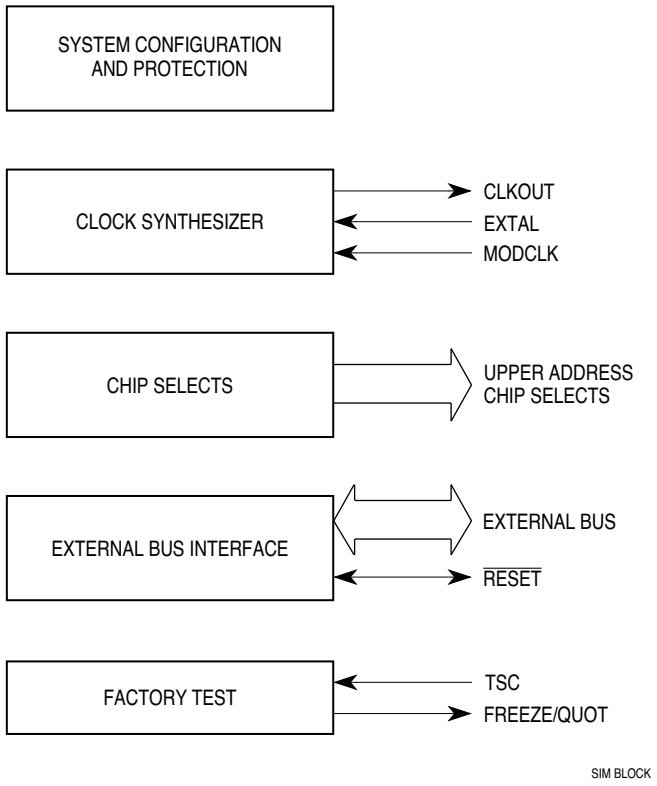


Figure 4 Single-Chip Integration Module Block Diagram

Table 7 SCIM Address Map

Address	15 8	7 0
YFFA00	SCIM MODULE CONFIGURATION (SCIMCR)	
YFFA02	FACTORY TEST (SCIMTR)	
YFFA04	CLOCK SYNTHESIZER CONTROL (SYNCR)	
YFFA06	UNUSED	RESET STATUS REGISTER (RSR)
YFFA08	MODULE TEST E (SCIMTRE)	
YFFA0A	PORT A DATA REGISTER (PORTA)	PORT B DATA REGISTER (PORTB)
YFFA0C	PORT G DATA REGISTER (PORTG)	PORT H DATA REGISTER (PORTH)
YFFA0E	PORT G DATA DIRECTION (DDRG)	PORT H DATA DIRECTION (DDRH)
YFFA10	UNUSED	PORTE DATA (PORTE0)
YFFA12	UNUSED	PORTE DATA (PORTE1)
YFFA14	PORT A/B DATA DIRECTION (DDRAB)	PORT E DATA DIRECTION (DDRE)
YFFA16	UNUSED	PORT E PIN ASSIGNMENT (PEPAR)
YFFA18	UNUSED	PORTF DATA (PORTF0)
YFFA1A	UNUSED	PORTF 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	PORTFE
YFFA2A	UNUSED	PORT F EDGE DETECT INTERRUPT (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 (PORTC)
YFFA42	UNUSED	UNUSED
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)	

Table 7 SCIM Address Map

Address	15 8	7 0
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 modmap bit in the SCIMCR.

3.1 System Configuration

The MC68HC16Y1 can operate as a stand-alone device (single-chip modes), with 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, since ADDR[23:20] follow the state of ADDR19, the external bus is effectively only 20 bits wide. In addition, 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 module configuration register (SCIMCR), and by asserting certain 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	2	1	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 — see bit description below

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 iscleared to zero when the MCU is in an expanded mode, and set 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 externally monitored. The table below 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

The SUPV bit places SCIM global registers in either supervisor data space or user data space. Since the CPU16 in the MC68HC16Y1 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 M determines the value of ADDR23 in the IMB module address. Because ADDR[23:20] follow the state of ADDR19 in the MC68HC16Y1, M must be set to one — if M is cleared, IMB modules will be 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 cleared to zero when the MCU is in an expanded mode, and set 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 cleared to zero when the MCU is in an expanded mode, and set 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.

RSR — Reset Status Register **$YFFA07**

7	6	5	4	3	2	1	0
EXT	POW	SW	HLT	0	LOC	SYS	TST

The reset status register contains a bit for each reset source in the MCU. A bit set to one 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 the CPU RESET instruction. System reset does not load a reset vector or affect any internal CPU registers or SIM configuration registers, but does reset external devices and other internal modules.

3.2 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.

Table 8 Basic Configuration Options

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

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

Operating mode determines which address and data bus lines are used and which general-purpose I/O ports are available. The table below summarizes bus and port configuration.

Table 9 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 MC68HC16, including data and address bus pins, have multiple functions. Reset value for these pins depends on operating mode. In expanded mode, the values of DATA[11:0] during reset determines the function of these pins. The functions of some pins can be changed subsequently by writing to the appropriate pin assignment register. Data bus pins have internal pull-ups and must be pulled low to achieve the desired alternate configuration. The following tables summarize pin configuration options for each operating mode.

3.2.1 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.

Table 10 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/$\overline{CS10}$/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]
$\overline{BGACK}$/$\overline{CSE}$ $\overline{BG}$/$\overline{CSM}$	DATA10	$\overline{BGACK}$ $\overline{BG}$	$\overline{CSE}$[1] $\overline{CSM}$[2]
DATA11	DATA11	Slave Mode Disabled[3]	Slave Mode Enabled[3]

1. $\overline{CSE}$ is enabled when DATA10 and DATA1 = 0 during reset.
2. $\overline{CSM}$ is enabled when DATA13, DATA10 and DATA1 = 0 during reset.
3. Slave mode used for factory test only.

3.2.2 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.

Table 11 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/$\overline{CS10}$/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 $\overline{SIZ1}$/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]
$\overline{BGACK}$/$\overline{CSE}$ $\overline{BG}$/$\overline{CSM}$	N/A[1]	$\overline{BGACK}$ $\overline{BG}$	$\overline{BGACK}$ $\overline{BG}$

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

3.2.3 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 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.

Table 12 Single-Chip Mode Reset Configuration

Pin(s) Affected	Function
$\overline{CSBOOT}$	$\overline{CSBOOT}$ 8-Bit
ADDR[18:10]	PA[7:0]
ADDR[9:3]	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 $\overline{AVEC}$/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]
$\overline{BGACK}$/$\overline{CSE}$ $\overline{BG}$/$\overline{CSM}$	$\overline{BGACK}$ $\overline{BG}$

3.3 Emulation Support

The SCIM contains logic that can be used to replace on-chip ports externally. It 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 does not respond to these accesses, allowing external logic, such as a port replacement unit (PRU) to respond. Accesses to externally-mapped registers require three clock cycles.

External ROM emulation is enabled by holding 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. The ROM module does not acknowledge IMB accesses while in emulation mode — this causes the SCIM to run an external bus cycle for each access. See **3.9 Chip Selects** and **9 Masked ROM Module** for more information.

3.3.1 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**

7	6	5	4	3	2	1	0
SWE	SWP	SWT		DBE	BME	BMT	
RESET:							
1	$\overline{\text{MODCLK}}$	0	0	0	0	0	0

The system protection control register controls system monitor functions, software watchdog clock prescaling, and bus monitor timing. In operating modes, this register can be written only once following power-on or reset, but can be read at any time. In test mode, it is writable 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 time-out period. The following table gives the ratio for each combination of SWP and SWT bits.

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

DBE — Double Bus Fault Enable

0 = Disable double bus fault halt monitor function
1 = Enable double bus fault 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 time-out period as shown in the table below.

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

3.3.2 Bus Monitor

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

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

3.3.3 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 DBE bit in the SYPCR.

3.3.4 Spurious Interrupt Monitor

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

3.3.5 Software Watchdog

SWSR — Software Service Register **$YFFA27**

7	6	5	4	3	2	1	0
SWSR							
RESET:							
0	0	0	0	0	0	0	0

Register shown with read value.

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

Perform a software watchdog service sequence as follows:

- Write $55 to SWSR.
- Write $AA to SWSR.

Both writes must occur in the order listed prior to time-out, 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 time-out period will take effect.

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

Software watchdog time-out period is given by the following equation:

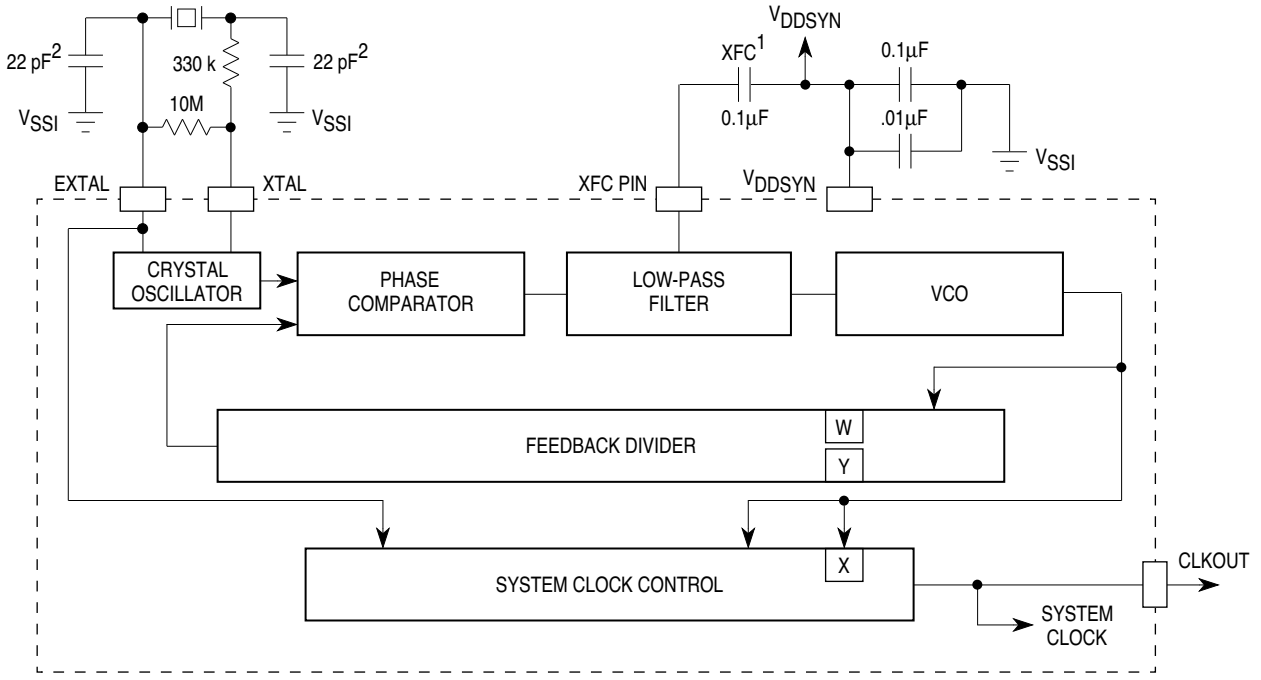
Time-Out Period = Divide Count/EXTAL Frequency

3.4 System Clock

The system clock in the SCIM provides timing signals for the IMB modules and for an external peripheral bus. Because the MC68HC16Y1 is a fully static design, register and memory contents are not affected when clock rate changes. System hardware and software support changes in clock rate during operation.

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

Following is a block diagram of the clock submodule.



1. Must be low-leakage capacitor (insulation resistance 30,000 MΩ or greater).
2. Capacitance based on a test circuit constructed with a DAISHINKU DMX-38 32.768-kHz crystal.

Figure 5 System Clock Block Diagram

3.4.1 Clock Sources

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

A reference crystal must be connected between the EXTAL and XTAL pins in order to use the internal oscillator. Use of a 32.768 kHz watch crystal is recommended — these crystals are readily available and inexpensive.

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

When an external system clock signal is applied (PLL not used), duty cycle of the input is critical, especially at operating frequencies close to maximum. The relationship between clock signal duty cycle and clock signal period is expressed:

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

3.4.2 Clock Synthesizer Operation

A voltage controlled oscillator (VCO) generates the system clock signal. A portion of the clock signal is fed back to a divider/counter. The divider controls the frequency of one input to a phase comparator. The other phase comparator input is a reference signal, either from the internal oscillator or from an external source. The comparator generates a control signal proportional to the difference in phase between its two inputs. The signal is low-pass filtered and used to correct VCO output frequency.

The synthesizer locks when VCO frequency is identical to reference frequency. Lock time is affected by the filter time constant and by the amount of difference between the two comparator inputs. Whenever comparator input changes, the synthesizer must re-lock. Lock status is shown by the SLOCK bit in SYNCR.

The MC68HC16Y1 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, connected between the XFC and V_{DDSYN} pins.

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

When the clock synthesizer is used, control register SYNCR determines operating frequency and various modes of operation. Because the CPU16 in the MC68HC16Y1 operates only in supervisor mode, SYNCR can be read or written at any time.

The SYNCR X bit controls a divide by two prescaler that is not in the synthesizer feedback loop. Setting X doubles clock speed without changing VCO speed — there is no VCO relock delay. The SYNCR W bit controls a 3-bit prescaler in the feedback divider. Setting W increases VCO speed by a factor of four.

The SYNCR Y field determines the count modulus for a modulo 64 down counter, causing it to divide by a value of Y + 1. When either W or Y value changes, there is a VCO relock delay.

Clock frequency is determined by SYNCR bit settings as follows:

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

In order for the device to perform correctly, the clock frequency selected by the W, X, and Y bits must be within the limits specified for the MCU. Maximum specified clock frequency with a 32.768 kHz reference is 16.78 kHz.

VCO frequency is determined by:

$$F_{VCO} = F_{SYSTEM}\ (2 - X),\ \text{for 32.768 kHz devices.}$$

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

3.4.3 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	12	11	10	9	8	7	6	5	4	3	2	1	0
W	X	Y						EDIV	0	0	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 status of or control 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 — E Clock Divide Rate

0 = ECLK frequency is system clock divided by 8.
1 = ECLK frequency is system clock divided by 16.

ECLK is an external M6800 bus clock available on pin ADDR23. See **3.9 Chip Selects** for more information.

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 will cause SLIMP to be set. The VCO continues to run using the base control voltage. Maximum limp frequency is 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 System Integration Clock

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

1 = When LPSTOP is executed, the SCIM clock is driven from the VCO.

STEXT — Stop Mode External Clock

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

1 = When LPSTOP is executed, the CLKOUT signal is driven from the SCIM clock, as determined by the state of the STSCIM bit.

3.4.4 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	6	5	4	3	2	1	0
0	0	0	0	0	PIRQL			PIV							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

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 table below 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	6	5	4	3	2	1	0
0	0	0	0	0	0	0	PTP	PITM							
RESET:															
0	0	0	0	0	0	0	$\overline{\text{MODCLK}}$	0	0	0	0	0	0	0	0

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:

$$\text{PIT Period} = [(\text{PITM})(\text{Prescale})(4)]/\text{EXTAL}$$

where

PIT Period = Periodic interrupt timer period

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

EXTAL = Crystal frequency

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

3.5 External Bus Interface

The external bus interface (EBI) transfers information between the internal MCU bus and external devices when the MC68HC16Y1 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 in the MC68HC16Y1 drives only 20 of the 24 IMB address lines, ADDR[23:20] follow the output state of 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. See **3.9 Chip Selects** for more information.

3.5.1 Bus Control Signals

The CPU initiates a bus cycle by driving the address, size, function code, and read/write outputs. At the beginning of the cycle, size signals SIZ0 and SIZ1 are driven along with the function code signals. The

size signals indicate the number of bytes remaining to be transferred during an operand cycle. They are valid while the address strobe ($\overline{AS}$) is asserted. The table below shows SIZ0 and SIZ1 encoding. The read/write (R/$\overline{W}$) signal determines the direction of the transfer during a bus cycle. This signal changes state, when required, at the beginning of a bus cycle, and is valid while $\overline{AS}$ is asserted. R/$\overline{W}$ only transitions when a write cycle is preceded by a read cycle or vice versa. The signal may remain low for two consecutive write cycles.

Table 13 Size Signal Encoding

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

3.5.2 Function Codes

Function code signals FC[2:0] are automatically generated by the CPU16. The function codes can be considered address extensions that automatically select one of eight address spaces to which an address applies. These spaces are designated as either user or supervisor, and program or data spaces. Because the CPU16 always operates in supervisor mode (FC2 always = 1), address spaces 0 to 3 are not used. Address space 7 is designated CPU space. CPU space is used for control information not normally associated with read or write bus cycles. Function codes are valid while $\overline{AS}$ is asserted.

Table 14 CPU16 Address Space Encoding

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

3.5.3 Address Bus

Address bus signals ADDR[19:0] define the address of the most significant byte to be transferred during a bus cycle. The MCU places the address on the bus at the beginning of a bus cycle. The address is valid while $\overline{AS}$ is asserted. Because the CPU16 in the MC68HC16Y1 does not drive ADDR[23:20], these lines follow the logic state of ADDR19.

3.5.4 Address Strobe

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

3.5.5 Data Bus

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

3.5.6 Data Strobe

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

3.5.7 Bus Cycle Termination Signals

During bus cycles, external devices assert the data transfer and size acknowledge signals ($\overline{\text{DSACK1}}$ and $\overline{\text{DSACK0}}$). 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 may terminate. 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 ($\overline{\text{BERR}}$) signal is also a bus cycle termination indicator and can be used in the absence of $\overline{\text{DSACK1}}$ and $\overline{\text{DSACK0}}$ 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 $\overline{\text{BERR}}$ signal for internal and internal-to-external transfers. When $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ are asserted simultaneously, the CPU16 takes a bus error exception.

Finally, autovector signal ($\overline{\text{AVEC}}$) can be used to terminate external IRQ pin interrupt acknowledge cycles. $\overline{\text{AVEC}}$ 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. $\overline{\text{AVEC}}$ is ignored during all other bus cycles.

3.5.8 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 (DSACK1and $\overline{\text{DSACK0}}$).

3.5.9 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 $\overline{\text{DSACK0}}$ and $\overline{\text{DSACK1}}$ inputs, as shown in the following table.

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

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

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 $\overline{\text{DSACK0}}$ and $\overline{\text{DSACK1}}$ signals to indicate the port width. For instance, a 16-bit device always returns $\overline{\text{DSACK0}}$ 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 figure below. 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	

Figure 6 Operand Byte Order

3.5.10 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] follow the state of ADDR19 in the MC68HC16Y1.

3.5.11 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.

In the MC68HC16Y1, 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 MC68HC11 CPU. The CPU16 treats misaligned long-word transfers as two misaligned word transfers.

3.5.12 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.

Table 16 Operand Transfer Cases

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

3.6 Reset

Reset procedures handle system initialization and recovery from catastrophic failure. The MC68HC16Y1 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.

SCIM Reset Mode Selection

The logic states of certain MCU pins during reset determine SCIM operating configuration. Refer to **3.2 Operating Modes** for more information.

3.6.1 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.

Table 17 Module Pin Functions

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

3.6.2 Reset Timing

The $\overline{RESET}$ input must be asserted for a specified minimum period in order for reset to occur. External $\overline{RESET}$ assertion can be delayed internally for a period equal to the longest bus cycle time (or the bus monitor time-out period) in order to protect write cycles from being aborted by reset. While $\overline{RESET}$ is asserted, SIM 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.

3.6.3 Power-On Reset

When the SCIM clock synthesizer is used to generate system clocks, power-on reset involves special circumstances related to application of system and clock synthesizer power. Regardless of clock source, voltage must be applied to clock synthesizer power input pin V_{DDSYN}, so that the MCU can op-

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

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

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

3.6.4 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-up reset:

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

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

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

3.7 Interrupts

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

The CPU16 provides for eight levels of interrupt priority (0–7), seven automatic interrupt vectors, and 200 assignable interrupt vectors. All interrupts with priorities less than seven 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 $\overline{\text{IRQ7}}$) from being recognized and processed. When IP contains %000, no interrupt is masked. During exception processing, the IP field is set to the priority of the interrupt being serviced.

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

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

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

$\overline{IRQ[6:1]}$ are maskable. $\overline{IRQ7}$ is nonmaskable. The $\overline{IRQ7}$ input is transition-sensitive 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.

3.7.1 Interrupt Acknowledge and Arbitration

Interrupt acknowledge bus cycles are generated during exception processing. When the CPU16 detects one or more interrupt requests of a priority higher than the interrupt priority mask value, it performs a CPU space read from address $FFFFF : [IP] : 1.

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

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

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

Because the EBI manages external interrupt requests, the SCIM IARB value is used for arbitration between internal and external interrupt requests. The reset value of IARB for the SCIM is %1111. The reset IARB value for all other modules is %0000. Initialization software must assign different IARB values to implement an arbitration scheme.

Each module must have a unique IARB value. When two or more IARB fields have the same nonzero value, the CPU16 interprets multiple vector numbers simultaneously, with unpredictable consequences.

Arbitration must always take place, even when a single source requests service. This point is important for two reasons: the CPU interrupt acknowledge cycle to 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 ac-

knowledge cycle is transferred to the external bus, an external device must decode the mask value and respond with a vector number, then generate bus cycle termination signals. If the device does not respond in time, a spurious interrupt exception is taken.

The periodic interrupt timer (PIT) in the SCIM can generate internal interrupt requests of specific priority at predetermined intervals. By hardware convention, PIT interrupts are serviced before external interrupt service requests of the same priority. Refer to **3.4.4 Periodic Interrupt Timer** for more information.

3.7.2 Interrupt Processing Summary

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

A. The CPU finishes higher priority exception processing or reaches an instruction boundary.
B. Processor state is stacked, then the CCR PK extension field is cleared.
C. The interrupt acknowledge cycle begins:
    1. FC[2:0] are driven to %111 (CPU space) encoding.
    2. The address bus is driven as follows. ADDR[23:20] = %1111; ADDR[19:16] = %1111, which indicates that the cycle is an interrupt acknowledge CPU space cycle; ADDR[15:4] = %111111111111; ADDR[3:1] = the priority of the interrupt request being acknowledged; and ADDR0 = %1.
    3. Request priority is latched into the CCR IP field from the address bus.
D. Modules or external peripherals that have requested interrupt service decode the priority value in ADDR[3:1]. If request priority is the same as the priority value in the address, IARB contention takes place. When there is no contention, the spurious interrupt monitor asserts $\overline{\text{BERR}}$, and a spurious interrupt exception is processed.
E. After arbitration, the interrupt acknowledge cycle can be completed in one of three ways:
    1. The dominant interrupt source supplies a vector number and $\overline{\text{DSACK}}$x signals appropriate to the access. The CPU16 acquires the vector number.
    2. The $\overline{\text{AVEC}}$ signal is asserted (the signal can be asserted by the dominant interrupt source or the pin can be tied low), and the CPU16 generates an autovector number corresponding to interrupt priority.
    3. The bus monitor asserts $\overline{\text{BERR}}$ and the CPU16 generates the spurious interrupt vector number.
F. The vector number is converted to a vector address.
G. The content of the vector address is loaded into the PC, and the processor transfers control to the exception handler routine.

3.8 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 which configure each pin as digital I/O or an alternate function. Port F has an edge-detect flag register which 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.

Table 18 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 port A, B, E, G, and H data and data direction registers and 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. Port F registers remain accessible.

A write to port A, B, E, F, G, or H data register is stored in the internal data latch, and 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.

3.8.1 Ports A and B

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

PORTA — Port A Data Register **$YFFA0A**

7	6	5	4	3	2	1	0
PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
RESET:							
U	U	U	U	U	U	U	U

PORTB — Port B Data Register **$YFFA0B**

7	6	5	4	3	2	1	0
PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
RESET:							
U	U	U	U	U	U	U	U

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

7	6	5	4	3	2	1	0
0	0	0	0	0	0	DDA	DDB
RESET:							
0	0	0	0	0	0	0	0

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.

3.8.2 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**

7	6	5	4	3	2	1	0
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**

7	6	5	4	3	2	1	0
DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0
RESET:							
0	0	0	0	0	0	0	0

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

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

7	6	5	4	3	2	1	0
PEPA7	PEPA6	PEPA5	PEPA4	PEPA3	PEPA2	PEPA1	PEPA0
RESET (Expanded, Single-chip):							
DATA8	DATA8	DATA8	DATA8	DATA8	DATA8	DATA8	DATA8
0	0	0	0	0	0	0	0

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

Table 19 Port E Pin Assignments

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

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

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

3.8.3 Port F

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

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

7	6	5	4	3	2	1	0
PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
RESET:							
U	U	U	U	U	U	U	U

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

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

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

7	6	5	4	3	2	1	0
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**

7	6	5	4	3	2	1	0
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 tables. $\overline{\text{BERR}}$ and DATA9 determine the reset state of this register. If $\overline{\text{BERR}}$ and/or DATA9 are low during reset, this register is set to $00, defining all port F pins to be I/O pins. If $\overline{\text{BERR}}$ and DATA9 are both high during reset, the register is set to $FF, which defines all port F pins except PF0 to be interrupt signals.

Table 20 Port F Pin Assignments

PFPAR Field	Port F Signal	Alternate Signal
PFPA3	PF[7:6]	$\overline{\text{IRQ}}$[7:6]
PFPA2	PF[5:4]	$\overline{\text{IRQ}}$[5:4]
PFPA1	PF[3:2]	$\overline{\text{IRQ}}$[3:2]
PFPA0	PF[1:0]	$\overline{\text{IRQ1}}$, MODCLK*

*MODCLK signal is only recognized during reset

Table 21 PFPAR Pin Functions

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

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

7	6	5	4	3	2	1	0
EF7	EF6	EF5	EF4	EF3	EF2	EF1	EF0
RESET:							
0	0	0	0	0	0	0	0

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

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

7	6	5	4	3	2	1	0
PFIVR7	PFIVR6	PFIVR5	PFIVR4	PFIVR3	PFIVR2	PFIVR1	PFIVR0
RESET:							
0	0	0	0	1	1	1	1

This register determines which vector in the exception vector table is used for interrupts generated by the port F edge-detect logic. Program PFIVR[7:0] to the value pointing to the appropriate interrupt vector. See the CPU16 section of this summary for interrupt vector assignments.

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

7	6	5	4	3	2	1	0
0	0	0	0	0	PFLV2	PFLV1	PFLV0
RESET:							
0	0	0	0	0	0	0	0

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

3.8.4 Port G

Port G is available in single-chip mode only. In single-chip mode, these pins are always configured for general-purpose I/O.

PORTG — Port G Data Register **$YFFA0C**

7	6	5	4	3	2	1	0
PG7	PG6	PG5	PG4	PG3	PG2	PG1	PG0
RESET:							
U	U	U	U	U	U	U	U

This register can be read or written anytime the MCU is not in emulator mode.

DDRG — Port G Data Direction Register **$YFFA0E**

7	6	5	4	3	2	1	0
DDG7	DDG6	DDG5	DDG4	DDG3	DDG2	DDG1	DDG0
RESET:							
0	0	0	0	0	0	0	0

The bits in this register control the direction of the pin drivers when the pins are configured as I/O. Any bit in this register set to one configures the corresponding pin as an output. Any bit in this register cleared to zero configures the corresponding pin as an input.

3.8.5 Port H

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

PORTH — Port H Data Register **$YFFA0D**

7	6	5	4	3	2	1	0
PH7	PH6	PH5	PH4	PH3	PH2	PH1	PH0
RESET:							
U	U	U	U	U	U	U	U

This register can be read or written anytime the MCU is not in emulator mode. Reset has no effect.

DDRH — Port H Data Direction Register **$YFFA0F**

7	6	5	4	3	2	1	0
DDH7	DDH6	DDH5	DDH4	DDH3	DDH2	DDH1	DDH0
RESET:							
0	0	0	0	0	0	0	0

The bits in this register control the direction of the pin drivers when the pins are configured as I/O. Any bit in this register set to one configures the corresponding pin as an output. Any bit in this register cleared to zero configures the corresponding pin as an input.

3.9 Chip Selects

Typical microcontrollers require additional hardware to provide external chip select signals. The MC68HC16Y1 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 emulator support. Address block sizes of 2 Kbytes to 1 Mbyte can be selected. However, because ADDR[23:20] = ADDR19 in the CPU16, 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 $\overline{\text{DSACK}}$ signals internally. A single $\overline{\text{DSACK}}$ 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, the appropriate chip select signal is asserted. Select signals are active low. A block diagram of a chip select circuit is shown below.

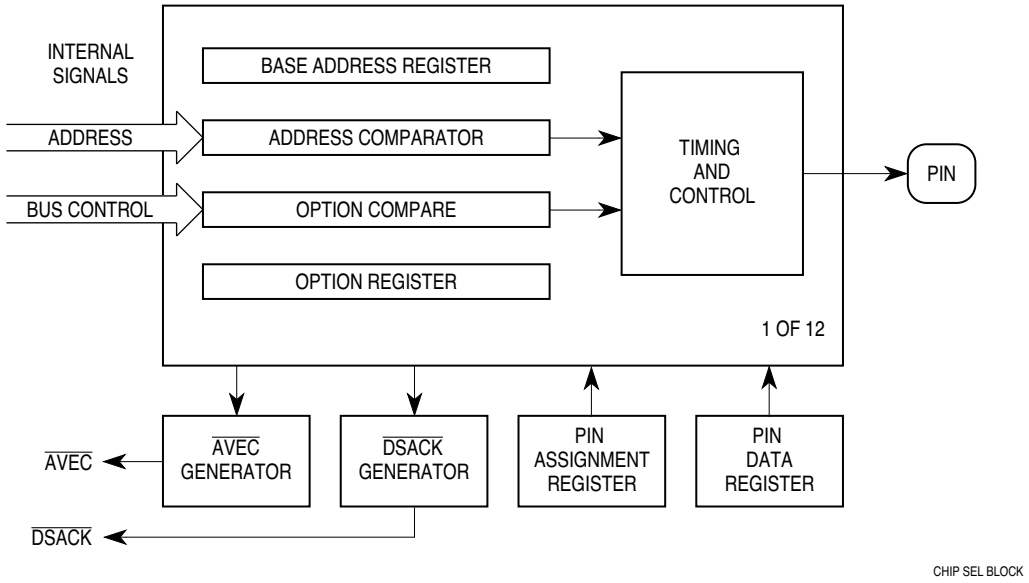


Figure 7 Chip Select Circuit Block Diagram

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

Each chip select pin can have two or more functions. Chip select 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 $\overline{\text{CS10}}$ and $\overline{\text{CS0}}$ 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.

Table 22 Chip Select Pin Allocation

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

3.10 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 $\overline{\text{CSE}}$ is asserted when any of these registers is addressed. The SCIM does not respond to these accesses — an external device, such as a port replacement unit, can respond instead. See **3.3 Emulation Support** for more information.

An internal module chip select signal $\overline{\text{CSM}}$ can also be enabled during emulator mode operation. When the ROM module is enabled, $\overline{\text{CSM}}$ is asserted when an access to an address assigned to the masked ROM array is made — this allows an external device to emulate the ROM. Internal $\overline{\text{DSACK}}$ is generated by the ROM module after it has inserted the number of wait states specified by the WAIT field in the MRMCR. See **9 Masked ROM Module** for more information.

3.10.1 Chip Select Registers

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

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

Blocks of addresses are assigned to each chip select function. Block sizes of 2 Kbytes to 1 Mbyte can be selected by writing values to the appropriate base address register (CSBAR). However, because the logic state of ADDR20 is always the same as the state of ADDR19 in the MC68HC16Y1, 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.

3.10.2 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 parentheses. 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	CS5 (FC2)		0	FC1	CS3 (FC0)		$\overline{\text{CSE}}$ (BGACK)		$\overline{\text{CSM}}$ (BG)		CS0 (BR)		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	CS10 (ADDR23)		CS9 (ADDR22)		CS8 (ADDR21)		CS7 (ADDR20)		CS6 (ADDR19)	
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. Clearing both CS10 select bits (CSPAR1[9:8]) enables the M6800 bus clock (ECLK) on ADDR23.

The table below shows pin assignment register encoding.

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

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

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

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

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

3.10.3 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

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

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 table below 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]

ADDR[23:20] will be at the same logic level as ADDR19 during normal operation.

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

3.10.4 Option Registers

The option registers contain eight fields that determine timing of and conditions for assertion of chip select signals. These make the chip selects useful for generating peripheral control signals. Certain constraints set by fields in the base address register and in the option register must be satisfied in order 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	8	7	6	5	4	3	2	1	0
MODE	BYTE		R/W		STRB	DSACK				SPACE		IPL			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	8	7	6	5	4	3	2	1	0
MODE	BYTE		R/W		STRB	DSACK				SPACE		IPL			AVEC
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

MODE — Asynchronous/Synchronous Mode

0 = Asynchronous mode selected

1 = Synchronous mode selected

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

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

BYTE — Upper/Lower Byte Option

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

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

R/W — Read/Write

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

The table below shows the options.

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

STRB — Address Strobe/Data Strobe

0 = Address strobe
1 = Data strobe

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

DSACK — Data 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 to optimize bus speed in a particular application. The following table shows the DSACK field encoding. A no-wait encoding (%0000) corresponds to a three clock-cycle bus. The fast termination encoding (%1110) corresponds to a two clock-cycle bus — microcontroller modules typically respond at this rate, but fast termination can also be used to access fast external memory.

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

SPACE — Address Space

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

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

IPL — Interrupt Priority Level

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

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

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

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

0 = External interrupt vector enabled

1 = Autovector enabled

This field selects one of two methods of acquiring the interrupt vector during the interrupt acknowledge cycle. It is not generally 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 $\overline{\text{AVEC}}$ field is set to one, the chip select automatically generates an $\overline{\text{AVEC}}$ in response to the interrupt acknowledge cycle. Otherwise, the vector must be supplied by the requesting device.

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

7	6	5	4	3	2	1	0
0	PC6	PC5	PC4	PC3	PC2	PC1	PC0
RESET:							
0	1	1	1	1	1	1	1

The pin data register controls 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. PC[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.

3.10.5 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 the discussion of these registers for more information.

The 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 eight 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 base address field in chip select base register boot (CSBARBT) has a reset value of all zeros. A ROM device containing vectors located at these addresses can be enabled by $\overline{\text{CSBOOT}}$ 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, since 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 $\overline{\text{CS-BOOT}}$.

Table 23 Chip Select Reset Values

Field	Reset Value
Base Address	$0000 0000
Block Size	512 Kbyte
Async/Sync Mode	Asynchronous Mode
Upper/Lower Byte	Both Bytes
Read/Write	Read/Write
AS/DS	AS
DSACK	13 Wait States
Address Space	Supervisor/User
IPL	All
Autovector	External Interrupt Vector

3.11 Factory Test

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

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

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

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

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

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

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

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

DREG — Distributed Register **$YFFA3A**

4 Time Processor Unit

The time processor unit (TPU) provides optimum performance in controlling time-related activity. The TPU contains a dedicated execution unit, a tri-level prioritized scheduler, data storage RAM, dual-time bases, and microcode ROM. The TPU controls 16 independent, orthogonal channels, each with an associated I/O pin, and is capable of performing any time function. Each channel also contains a dedicated event register, allowing both match and input capture functions. A block diagram of the TPU follows.

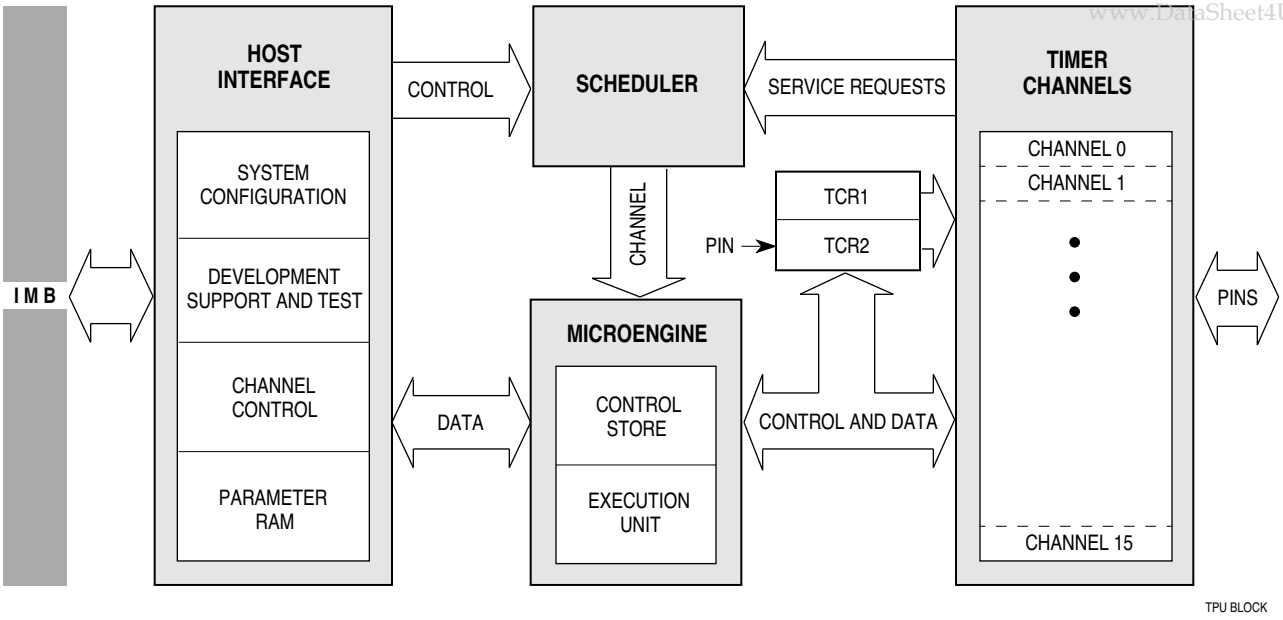


Figure 8 TPU Simplified Block Diagram

Table 24 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 is the state of the MODMAP bit in the SCIMCR (Y = $7 or $F)

4.1 TPU ROM Functions

4.1.1 Discrete Input/Output (DIO)

When a pin is used as a discrete input, a parameter indicates the current input level and the previous 15 levels of a pin. Bit 15, the most significant bit of the parameter, indicates the most recent state. Bit 14 indicates the next most recent state, and so on.

4.1.2 Input Capture/Input Transition Counter (ITC)

Any channel of the TPU can capture the value of a specified TCR upon the occurrence of each transition, and then generate an interrupt request to notify the bus master.

4.1.3 Output Compare (OC)

Generates a rising edge, a falling edge, or a toggle of the previous edge in either of two ways:

1. At a user-specified time. The CPU can also force an immediate output, thereby generating a pulse with a length equal to the programmed delay.
2. When linked to another channel. The OC references a linked parameter, without CPU interaction, and adds an offset to it.

4.1.4 Pulse-Width Modulation (PWM)

Generates a pulse-width modulated waveform with any duty cycle from 0% to 100% (within the resolution and latency capability of the TPU). The CPU provides one parameter that specifies waveform period and another parameter that specifies waveform high time. Updates to one or both of these parameters can change the output to take effect either immediately, or coherently, at the next low-to-high transition of the pin.

4.1.5 Synchronized Pulse-Width Modulation (SPWM)

Generates a pulse-width modulated waveform. The CPU can change period and/or high time at any time. When synchronized to a time function on a second channel, the SPWM low-to-high transitions have a time relationship to transitions on the second channel.

4.1.6 Period Measurement with Additional Transition Detect (PMA)

Allows special-purpose 16-bit period measurement. Detects the occurrence of an additional transition indicated by the current measurement being less than a programmed ratio of a previous measurement. When detected, this condition can be counted and compared to a programmed number of additional transitions.

4.1.7 Period Measurement with Missing Transition Detect (PMM)

Allows special-purpose 16-bit period measurement. Detects the occurrence of a missing transition indicated by the current measurement being more than a previous measurement multiplied by a programmed ratio. When detected, this condition can be counted and compared to a programmed number of transitions.

4.1.8 Position-Synchronized Pulse Generator (PSP)

Any channel of the TPU can generate an output transition or pulse, which is a projection in time based on a reference period previously calculated on another channel.

4.1.9 Stepper Motor (SM)

The stepper motor control algorithm uses a programmable number of step rates to control the linear acceleration and deceleration of a stepper motor. Any group of up to eight channels can be programmed to generate the control logic necessary to drive a stepper motor. Nominally, only two or four channels are used for a two-phase motor.

4.1.10 Period/Pulse-Width Accumulator (PPWA)

The PPWA continuously accumulates the high time or the total elapsed interval of a waveform over a programmed number of input periods. It continuously tracks current and most recent accumulated times.

Table 25 Parameter RAM Map

Channel	Parameter							
	0	1	2	3	4	5	6	7
0	X $YFFF00	02	04	06	08	0A	—	—
1	X $YFFF10	12	14	16	18	1A	—	—
2	X $YFFF20	22	24	26	28	2A	—	—
3	X $YFFF30	32	34	36	38	3A	—	—
4	X $YFFF40	42	44	46	48	4A	—	—
5	X $YFFF50	52	54	56	58	5A	—	—
6	X $YFFF60	62	64	66	68	6A	—	—
7	X $YFFF70	72	74	76	78	7A	—	—
8	X $YFFF80	82	84	86	88	8A	—	—
9	X $YFFF90	92	94	96	98	9A	—	—
10	X $YFFFA0	A2	A4	A6	A8	AA	—	—
11	X $YFFFB0	B2	B4	B6	B8	BA	—	—
12	X $YFFFC0	C2	C4	C6	C8	CA	—	—
13	X $YFFFD0	D2	D4	D6	D8	DA	—	—
14	X $YFFFE0	E2	E4	E6	E8	EA	EC	EE
15	X $YFFFF0	F2	F4	F6	F8	FA	FC	FE

— = Not Implemented

X = Assignable as supervisor accessible only (if SUPV = 1) or unrestricted (if SUPV = 0). Unrestricted allows both user and supervisor access.

Y = M111, where M is the modmap bit in the module configuration register of the single-chip integration module (Y = $7 or $F).

4.2 TPU Registers

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

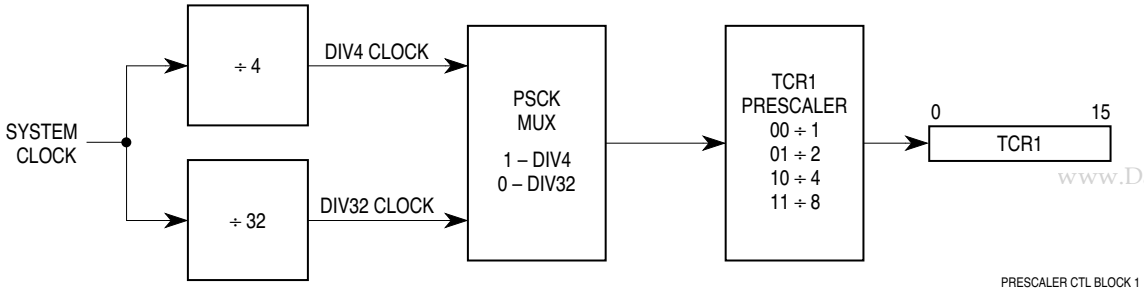
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP	TCR1P		TCR2P		EMU	T2CG	STF	SUPV	PSCK	0	0	IARB			
RESET:															
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

STOP — Stop Bit

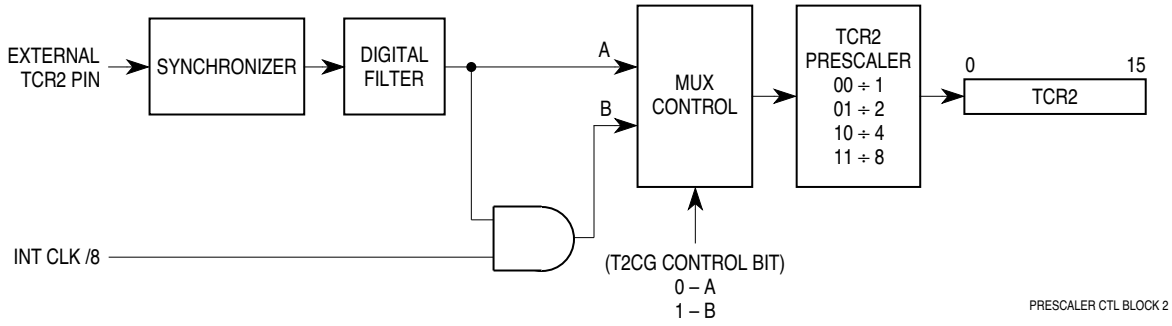
0 = Internal clocks not shut down (reset condition)

1 = Internal clocks shut down

TCR1P — TCR1 Prescaler Control



TCR2P — TCR2 Prescaler Control



EMU — Emulation Control

0 = TPU and RAM not in emulation mode (reset condition)
1 = TPU and RAM in emulation mode

T2CG — TCR2 Clock/Gate Control

0 = TCR2 pin used as clock source for TCR2 (reset condition)
1 = TCR2 pin used as gate of DIV8 clock for TCR2

STF — Stop Flag

0 = TPU operating (reset condition)
1 = TPU stopped

SUPV — Supervisor Data Space

0 = Assignable registers are unrestricted (FC2 is ignored).
1 = Assignable registers are restricted (FC2 is decoded; reset condition).

PSCK — Prescaler Clock

0 = DIV32 (system clock/32) is input to TCR1 prescaler.
1 = DIV4 (system clock/4) is input to TCR1 prescaler.

IARB — Interrupt Arbitration ID Bits

Each module that generates interrupts has an IARB field. The value in this field is used to arbitrate between simultaneous interrupt requests of the same priority. The reset value of all IARB fields other than that of the SCIM is $0 (lowest priority), to prevent priority conflict during initialization. The IARB field must be initialized to a value between $F (highest priority) and $1 (lowest priority), or subsequent interrupt requests will be identified by the CPU as spurious.

TCR — Test Configuration Register **$YFFE02**

This register is used for Motorola factory test only.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HOT4	0	0	0	0	BLC	CLKS	FRZ		CCL	BP	BC	BH	BL	BM	BT
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HOT4 — Hang on T4

0 = Exit wait on T4 state caused by assertion of HOT4

1 = Enter wait on T4 state

DSCR[14:11] — Not Implemented

BLC — Branch Latch Control

1 = Do not latch conditions into branch condition register before exiting the halted state or during the time-slot transition period.

0 = Latch conditions into branch condition register prior to exiting halted state.

CLKS — Stop Clocks (to TCRs)

0 = Do not stop TCRs.

1 = Stop TCRs during the halted state.

FRZ[1:0] — IMB FREEZE Response

The FRZ bits specify the TPU microengine response to the FREEZE signal.

FRZ[1:0]	TPU Response
00	Ignore Freeze
01	Reserved
10	Freeze at End of Current Microcycle
11	Freeze at Next Time-Slot Boundary

CCL — Channel Conditions Latch

CCL controls the latching of channel conditions (MRL and TDL) when CHAN is written.

0 = Only the pin state condition of the new channel is latched as a result of the write CHAN register microinstruction.

1 = Pin state, MRL, and TDL conditions of the new channel are latched as a result of a write CHAN register microinstruction.

BP, BC, BH, BL, BM, and BT — Breakpoint Enable Bits

DSCR[5:0] are TPU breakpoint enables. Setting a bit enables a breakpoint condition.

BP —Break if µPC equals µPC breakpoint register.

BC —Break if CHAN register equals channel breakpoint register at beginning of state or when CHAN is changed through microcode.

BH —Break if host service latch is asserted at beginning of state.

BL —Break if link service latch is asserted at beginning of state.

BM —Break if MRL is asserted at beginning of state.

BT —Break if TDL is asserted at beginning of state.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	BKPT	PCBK	CHBK	SRBK	TPUF	0	0	0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSSR[15:8, 2:0] — Not Implemented

BKPT — Breakpoint Asserted Flag

If an internal breakpoint caused the TPU to enter the halted state, the TPU asserts the BKPT signal on the IMB and the BKPT flag. The TPU continues to assert BKPT until it recognizes a breakpoint acknowledge cycle from a host, or until the FREEZE signal on the IMB is asserted.

PCBK — μPC Breakpoint Flag

PCBK is asserted if a breakpoint occurs because of a μPC register match with the μPC breakpoint register. PCBK is negated when the BKPT flag is negated.

CHBK — Channel Register Breakpoint Flag

CHBK is asserted if a breakpoint occurs because of a CHAN register match with the channel register breakpoint register. CHBK is negated when the BKPT flag is negated.

SRBK — Service Request Breakpoint Flag

SRBK is asserted if a breakpoint occurs because of any of the service request latches being asserted along with their corresponding enable flag in the development support control register. SRBK is negated when the BKPT flag is negated.

TPUF — TPU FREEZE Flag

TPUF is asserted whenever the TPU is in a halted state as a result of FREEZE being asserted. This flag is automatically negated when the TPU exits the halted state because of FREEZE being negated.

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

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

TICR[15:11] — Not Implemented

CIRL — Channel Interrupt Request Level

The interrupt request level for all channels is specified by this three-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

This field specifies the most significant nibble of all 16 TPU channel interrupt vector numbers.

TICR[3:0] — Not Implemented

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] — Interrupt Enable/Disable for each Channel

0 = Channel interrupts disabled
1 = Channel interrupts enabled

CFSR0–CFSR3 — Channel Function Select Registers **$YFFE0C–$YFFE12**

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

CFSR[15:0] — Encoded One of 16 Time Functions for each Channel

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

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

CHX[1:0]	Service
00	No Host Service (Reset Condition)
01	Type 1 Host Service
10	Type 2 Host Service
11	Type 3 Host Service

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
00	Disabled
01	Low
10	Middle
11	High

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] — Interrupt Status Bit
0 = Channel interrupt not asserted
1 = Channel interrupt asserted

Table 26 Host Service Summary

Function Name	Function Code	Host Service Request Code	Host Sequence Code*
DIO Discrete Input/ Output	$8	1= Force Output High 2 = Force Output Low 3 = Initialization, Input Specified 3 = Initialization, Periodic Input 3 = Update Pin Status Parameter	0 = Trans Mode — Record Pin on Transition 0 = Trans Mode — Record Pin on Transition 0 = Trans Mode — Record Pin on Transition 1 = Match Mode — Record Pin at Match_Rate 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

*Host Sequence Code interpretation is determined by the function; some HSQ codes apply to all HSR codes, some to only one, such as Init.

LR — Link Register **$YFFE22**

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

CH[15:0] — Test Mode Link Service Request Enable Bit
0 = Link bit not asserted
1 = Link bit asserted

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

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

CH[15:0] — Service Granted Bits

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

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

CH[15:0] — Service Status Bits

5 General-Purpose Timer Module

The GPT is a simple, yet flexible 11-channel timer for use in systems where a moderate degree of external visibility and control is required. The GPT consists of two nearly independent submodules, the compare/capture unit, and the pulse-width modulator. A block diagram of the GPT appears below.

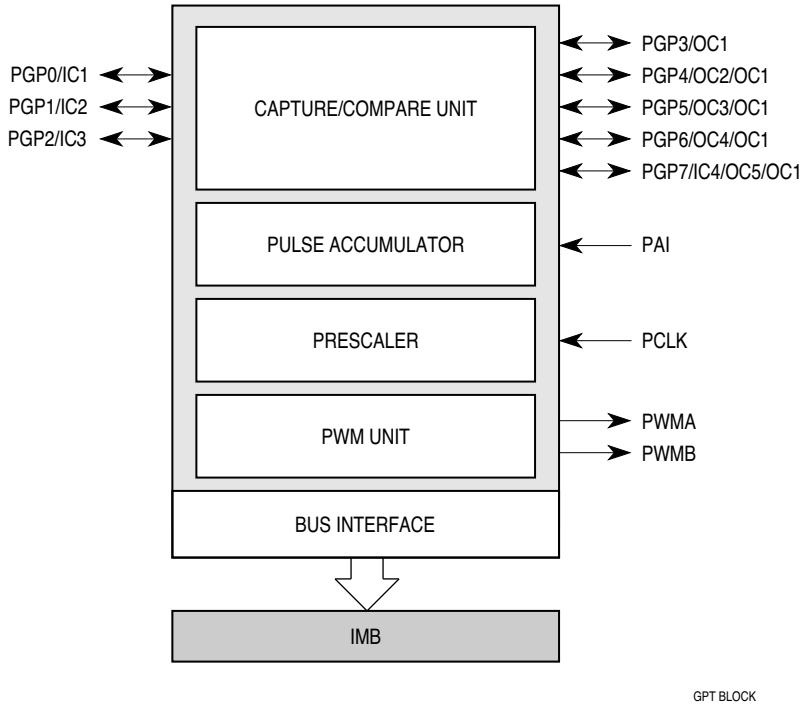


Figure 9 GPT Block Diagram

GPT IC/OC pins are bidirectional, and may be used to form an 8-bit parallel port. Pulse-width modulator outputs can also be used as general-purpose outputs. PAI and PCLK inputs can also be used as general-purpose inputs.

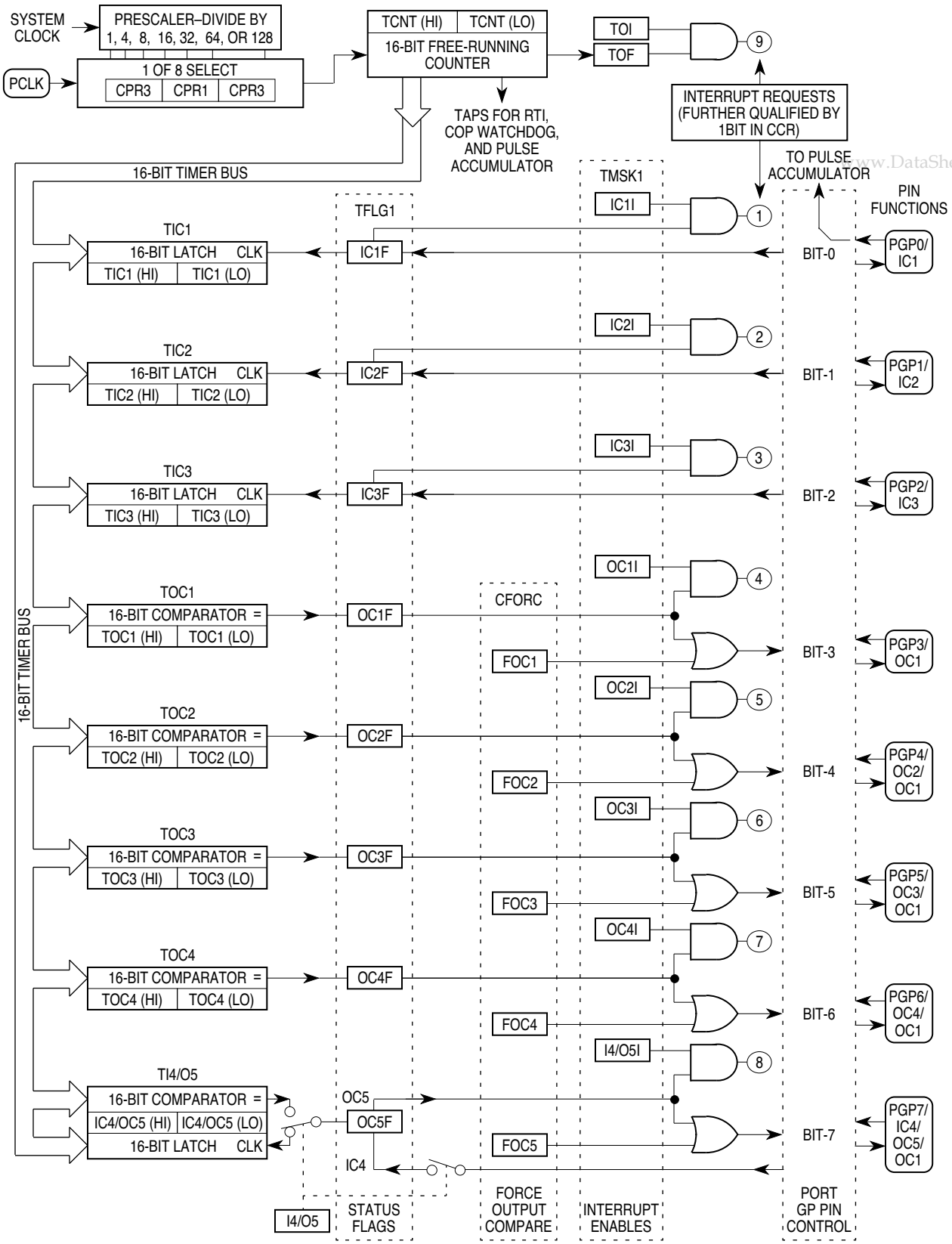
Table 27 GPT Address Map

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 state of the modmap bit in the module configuration register of the single-chip integration module. In an MC68HC16Y1 system, M must always be set to one.

5.1 Compare/Capture Unit

The compare/capture unit features three input capture channels, four output compare channels, and one input capture/output compare channel (function selected via control register). These channels share a 16-bit free-running counter (TCNT) which derives its clock from seven stages of a 9-stage prescaler or from external clock input PCLK. 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. Block diagrams of the GPT timer and prescaler follow.



NOTE: Parallel port pin functions are controlled by PDDR, OC1M, OC1D, and TCTL1 register.

GPT TIMER BLOCK

Figure 10 GPT Timer Diagram

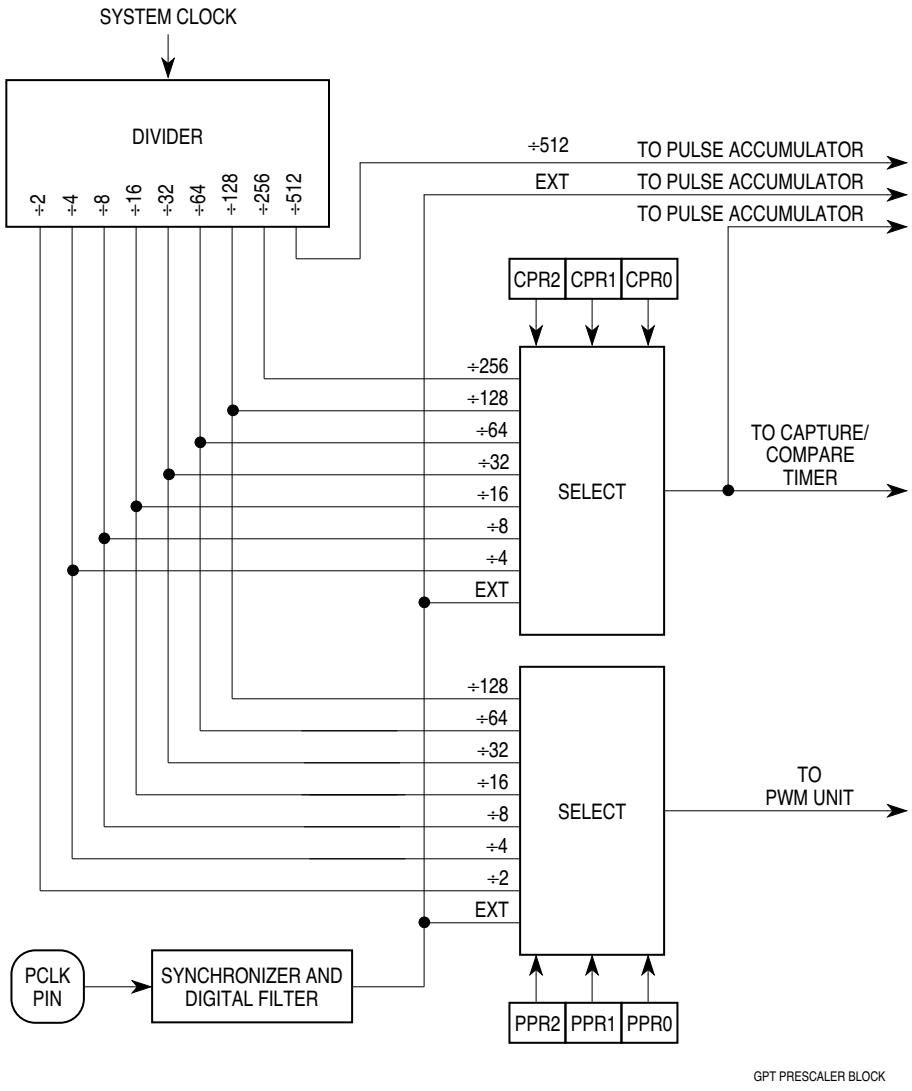


Figure 11 Prescaler Block Diagram

5.2 Pulse-Width Modulator

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

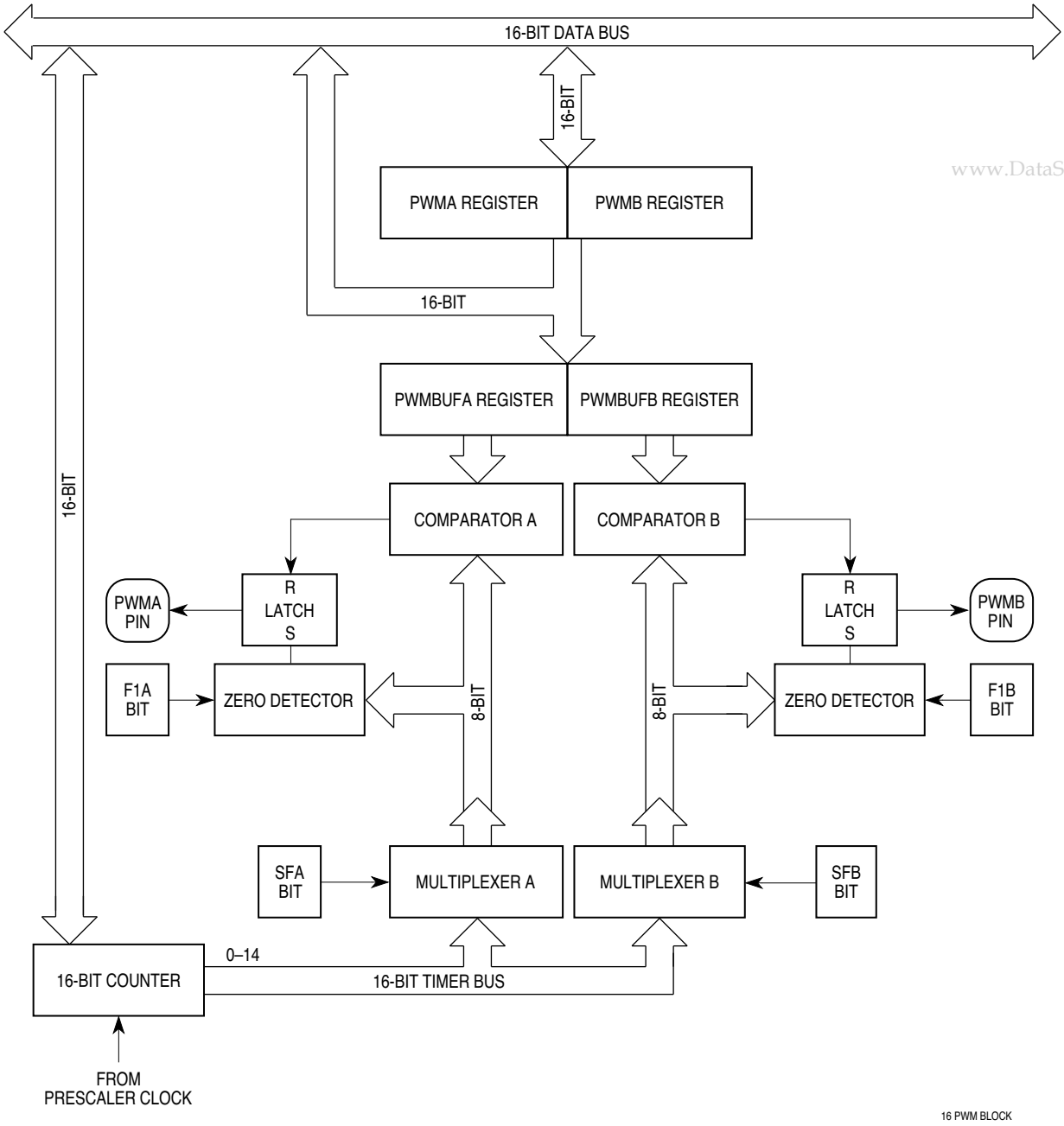


Figure 12 PWM Unit Block Diagram

5.3 GPT Registers

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP	FRZ		STOPP	INCP	0	0	0	SUPV	0	0	0	IARB			
RESET:															
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

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

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

FRZ1 — FREEZE Response
Reserved; has no effect

FRZ0 — FREEZE Response
- 0 = Ignore FREEZE
- 1 = FREEZE the current state of the GPT

STOPP — Stop Prescaler
- 0 = Normal operation
- 1 = Stop prescaler and pulse accumulator from incrementing. Ignore changes to input pins.

INCP — Increment Prescaler
- 0 = Has no meaning
- 1 = If STOPP is asserted, increment prescaler once and clock input synchronizers once.

SUPV — Supervisor/Unrestricted Data Space
- 0 = Registers with access controlled by SUPV are unrestricted (FC2 is a don't care).
- 1 = Registers with access controlled by SUPV are restricted when FC2 = 1.

Because the CPU16 in the MC68HC16Y1 operates in supervisor mode only (FC2 is always logic level one), this bit has no effect.

IARB[3:0] — Interrupt Arbitration ID

Each module that generates interrupts has an IARB field. The value in this field is used to arbitrate between simultaneous interrupt requests of the same priority. The reset value of all IARB fields other than that of the SCIM is $0 (lowest priority), to prevent priority conflict during initialization. The IARB field must be initialized to a value between $F (highest priority) and $1 (lowest priority), or subsequent interrupt requests will be identified by the CPU as spurious.

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

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

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIORITY ADJUST				0	INT REQUEST LEVEL			VECTOR BASE ADDRESS				0	0	0	0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Priority Adjust Field — This field specifies an interrupt to be advanced to the highest priority.

Interrupt Request Level — This field specifies the priority level of interrupts generated by the GPT.

Vector Base Address — This is the most significant nibble of interrupt vectors generated by the GPT.

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

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

When GPT pins are used as an 8-bit port, DDRGP determines whether pins are input or output and PORTGP holds the 8-bit data.

DDRGP[7:0] — Port GP Data Direction Register

0 = Input only

1 = Output

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

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0C1M					0	0	0	0C1D					0	0	0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

All OC outputs can be controlled by the action of OC1. OC1M contains a mask that determines which pins are affected, and OC1D determines what the outputs are.

OC1M[5:1] — OC1 Mask Field

0 = Corresponding output compare pin is not affected by OC1 compare.

1 = Corresponding output compare pin is affected by OC1 compare.

OC1M[5:1] correspond to OC[5:1].

OC1D[5:1] — OC1 Data Field

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

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

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

TCNT — Timer Counter Register **$YFF90A**

TCNT is the 16-bit free-running counter associated with the input capture, output compare, and pulse accumulator functions of the GPT module.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAIS	PAEN	PAMOD	PEDGE	PCLKS	I4/O5	PACLK		PULSE ACCUMULATOR COUNTER							
RESET:															
U	0	0	0	U	0	0	0	0	0	0	0	0	0	0	0

PACTL enables the pulse accumulator and selects either event counting or gated mode. In event counting mode, PACNT is incremented each time an event occurs. In gated mode, it is incremented by an internal clock.

PAIS — PAI Pin State (Read-Only)

PAEN — Pulse Accumulator System Enable

0 = Pulse accumulator disabled

1 = Pulse accumulator enabled

PAMOD — Pulse Accumulator Mode
0 = External event counting
1 = Gated time accumulation

PEDGE — Pulse Accumulator Edge Control
The effects of PEDGE and PAMOD are shown in the following table.

PAMOD	PEDGE	Effect
0	0	PAI Falling Edge Increments Counter
0	1	PAI Rising Edge Increments Counter
1	0	Zero on PAI Inhibits Counting
1	1	One on PAI Inhibits Counting

PCLKS — PCLK Pin State (Read-Only)

I4/O5 — Input Capture 4/Output Compare 5
0 = Output compare 5 enabled
1 = Input capture 4 enabled

PACLK[1:0] — Pulse Accumulator Clock Select (Gated Mode)

PACLK[1:0]	Pulse Accumulator Clock Selected
00	System Clock Divided by 512
01	Same Clock Used to Increment TCNT
10	TOF Flag from TCNT
11	External Clock, PCLK

PACNT — Pulse Accumulator Counter
Eight-bit read/write counter used for external event counting or gated time accumulation.

TIC1–TIC3 — Input Capture Registers 1–3 **$YFF90E, $YFF910, $YFF912**

The input capture registers are 16-bit read-only registers which are used to latch the value of TCNT when a specified transition is detected on the corresponding input capture pin. They are reset to $FFFF.

TOC1–TOC4 — Output Compare Registers 1–4 **$YFF914, $YFF916, $YFF918, $YFF91A**

The output compare registers are 16-bit read/write registers which can be used as output waveform controls or as elapsed time indicators. For output compare functions, they are written to a desired match value and compared against TCNT to control specified pin actions. They are reset to $FFFF.

TI4/O5 — Input Capture 4/Output Compare 5 Register **$YFF91C**

This register serves either as input capture register 4 or output compare register 5, depending on the state of I4/O5 in PACTL.

TCTL1/TCTL2 — Timer Control Registers 1–2 **$YFF91E**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OM5	OL5	OM4	OL4	OM3	OL3	OM2	OL2	EDGE4		EDGE3		EDGE2		EDGE1	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TCTL1 determines output compare mode and output logic level. TCTL2 determines the type of input capture to be performed.

OM/OL[5:2] — Output Compare Mode Bits and Output Compare Level Bits
Each pair of bits specifies an action to be taken when output comparison is successful.

OM/OL[5:2]	Action Taken
00	Timer Disconnected from Output Logic
01	Toggle OCx Output Line
10	Clear OCx Output Line to zero
11	Set OCx Output Line to one

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

TMSK1/TMSK2 — Timer Interrupt Mask Registers 1–2 **$YFF920**

15	14	13	12	11	10	9	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

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

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].

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

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]	Prescaler Value
000	4
001	8
010	16
011	32
100	64
101	128
110	256
111	PCLK

TFLG1/TFLG2 — Timer Interrupt Flag Registers 1–2 **$YFF922**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I4/O5F	OCF				ICF			TOF	0	PAOVF	PAIF	0	0	0	0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

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].

I4/O5F — Input Capture 4/Output Compare 5 Flag

When I4/O5 in PACTL is 0, this flag is set each time TCNT matches the value in TOC5. When I4/O5 in PACTL is 1, the flag is set each time a selected edge is detected at the I4/O5 pin.

TOF — Timer Overflow Flag

This flag is set each time TCNT advances from a value of $FFFF to $0000.

PAOVF — Pulse Accumulator Overflow Flag

This flag is set each time the pulse accumulator counter advances from a value of $FF to $00.

PAIF — Pulse Accumulator Flag

In event counting mode, this flag is set when an active edge is detected on the PAI pin. In gated time accumulation mode, it is set at the end of the timed period.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FOC					0	FPWMA	FPWMB	PPROUT	PPR			SFA	SFB	F1A	F1B
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

FOC[5:1] — Force Output Compare
0 = Has no meaning
1 = Causes pin action programmed for corresponding OC pin, but the OC Flag is not set.
FOC[5:1] correspond to OC[5:1].

FPWMA — Force PWMA Value
0 = Normal PWMA operation
1 = The value of F1A is driven out on the PWMA pin, regardless of the state of PPROUT.

FPWMB — Force PWMB Value
0 = Normal PWMB operation
1 = The value of F1B is driven out on the PWMB pin.

PPROUT — PWM Clock Output Enable
0 = Normal PWM operation on PWMA
1 = TCNT clock driven out PWMA pin

PPR[2:0] — PWM Prescaler/PCLK Select
This field selects one of seven prescaler taps or PCLK to be PWMCNT input.

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

SFA — PWMA Slow/Fast Select
0 = PWMA period is 256 PWMCNT increments long.
1 = PWMA period is 32768 PWMCNT increments long.

SFB — PWMB Slow/Fast Select
0 = PWMB period is 256 PWMCNT increments long.
1 = PWMB period is 32768 PWMCNT increments long.
The following table shows the effects of SF settings on PWM frequency (16.78-MHz system clock).

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

F1A — Force Logic Level on PWMA
0 = Force logic level zero output on PWMA pin.
1 = Force logic level one output on PWMA pin.

F1B — Force Logic Level on PWMB
0 = Force logic level zero output on PWMB pin.
1 = Force logic level one output on PWMB pin.

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

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

PWMCNT — PWM Count Register **$YFF928**

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

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

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

PRESCL — GPT Prescaler **$YFF92C**

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

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. Accuracy is ±1 count (one LSB) in 8-bit mode and ± 4 counts (two LSB) in 10-bit mode. Monotonicity is guaranteed in both modes. The ADC can perform an 8-bit single conversion (4-clock sample) in ten microseconds; a 10-bit single conversion in 11 microseconds. The following table is a module address map.

Table 28 ADC Module Address Map

Address	15 8	7 0
$YFF700	MODULE CONFIGURATION (ADCMCR)	
$YFF702	FACTORY TEST (ADTEST)	
$YFF704	(RESERVED)	
$YFF706	PORT ADA DATA (PORTADA)	
$YFF708	(RESERVED)	
$YFF70A	ADC CONTROL 0 (ADCTL0)	
$YFF70C	ADC CONTROL 1 (ADCTL1)	
$YFF70E	ADC STATUS (ADSTAT)	
$YFF710	RIGHT-JUSTIFIED UNSIGNED RESULT 0 (RJURR0)	
$YFF712	RIGHT-JUSTIFIED UNSIGNED RESULT 1 (RJURR1)	
$YFF714	RIGHT-JUSTIFIED UNSIGNED RESULT 2 (RJURR2)	
$YFF716	RIGHT-JUSTIFIED UNSIGNED RESULT 3 (RJURR3)	
$YFF718	RIGHT-JUSTIFIED UNSIGNED RESULT 4 (RJURR4)	
$YFF71A	RIGHT-JUSTIFIED UNSIGNED RESULT 5 (RJURR5)	
$YFF71C	RIGHT-JUSTIFIED UNSIGNED RESULT 6 (RJURR6)	
$YFF71E	RIGHT-JUSTIFIED UNSIGNED RESULT 7 (RJURR7)	
$YFF720	LEFT-JUSTIFIED SIGNED RESULT 0 (LJSRR0)	
$YFF722	LEFT-JUSTIFIED SIGNED RESULT 1 (LJSRR1)	
$YFF724	LEFT-JUSTIFIED SIGNED RESULT 2 (LJSRR2)	
$YFF726	LEFT-JUSTIFIED SIGNED RESULT 3 (LJSRR3)	
$YFF728	LEFT-JUSTIFIED SIGNED RESULT 4 (LJSRR4)	
$YFF72A	LEFT-JUSTIFIED SIGNED RESULT 5 (LJSRR5)	
$YFF72C	LEFT-JUSTIFIED SIGNED RESULT 6 (LJSRR6)	
$YFF72E	LEFT-JUSTIFIED SIGNED RESULT 7 (LJSRR7)	
$YFF730	LEFT-JUSTIFIED UNSIGNED RESULT 0 (LJURR0)	
$YFF732	LEFT-JUSTIFIED UNSIGNED RESULT 1 (LJURR1)	
$YFF734	LEFT-JUSTIFIED UNSIGNED RESULT 2 (LJURR2)	
$YFF736	LEFT-JUSTIFIED UNSIGNED RESULT 3 (LJURR3)	
$YFF738	LEFT-JUSTIFIED UNSIGNED RESULT 4 (LJURR4)	
$YFF73A	LEFT-JUSTIFIED UNSIGNED RESULT 5 (LJURR5)	
$YFF73C	LEFT-JUSTIFIED UNSIGNED RESULT 6 (LJURR6)	
$YFF73E	LEFT-JUSTIFIED UNSIGNED RESULT 7 (LJURR7)	

Y = M111, where M is the state of the modmap bit in the SCIMCR. In the MC68HC16Y1, Y must equal $F — if M is cleared, IMB modules will be inaccessible until a reset occurs. M can be written only once after reset.

6.1 ADC Operation

ADC functions can be grouped into three basic subsystems: an analog front end, a digital control section, and a bus interface. A block diagram of the converter follows.

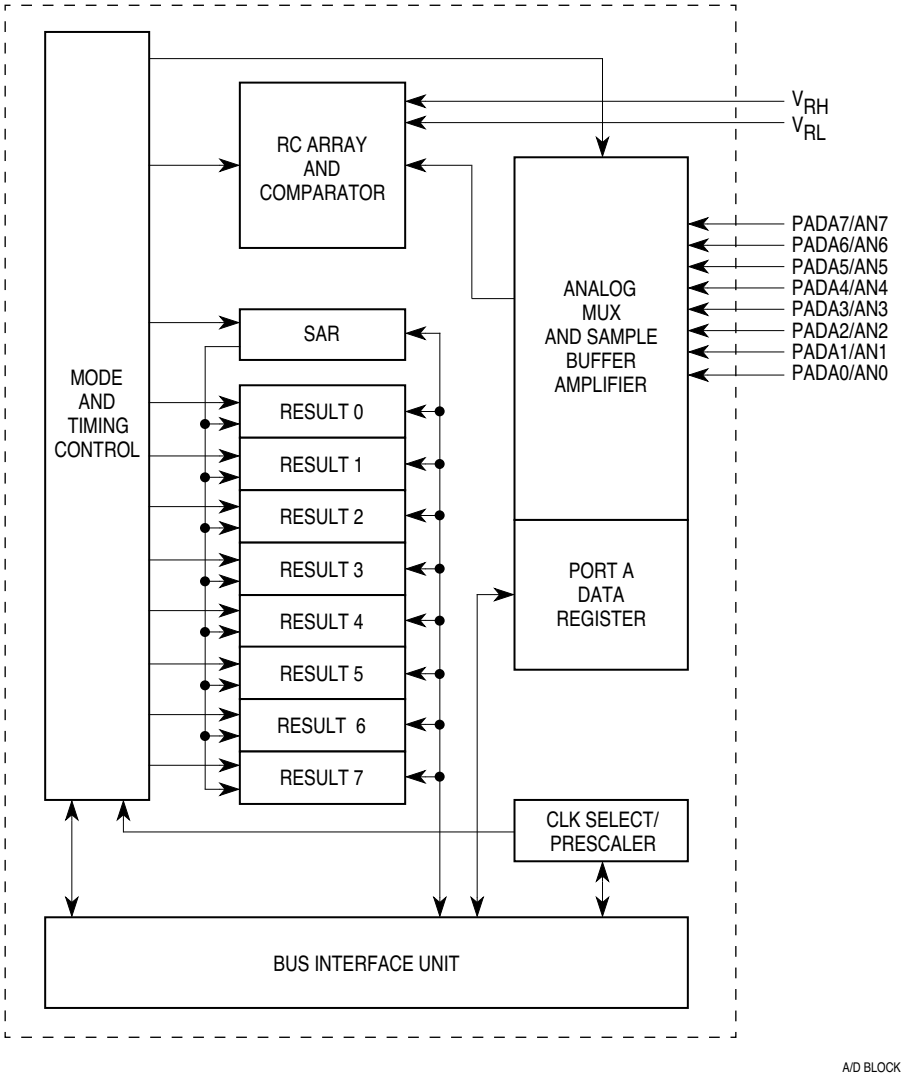


Figure 13 Analog-to-Digital Converter Block Diagram

6.2 Analog Subsystem

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

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 will abort any conversion in progress. STOP is set to logic level one at reset, and may be cleared to logic level zero by the CPU.

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

FRZ[1:0] — Freeze 1

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

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

SUPV — Supervisor/Unrestricted

0 = Unrestricted access
1 = Supervisor access

SUPV defines access to assignable ADC registers. Because the CPU16 in the MC68HC16Y1 operates in supervisor mode only, this bit has no effect.

ADTEST — ADC Test Register **$YFF702**

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

PORTADA — Port Data Register **$YFF706**

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

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

Port ADA Data[7:0]

A read of PORTADA[7:0] will return the logic level of the port A pins. If the input is not an appropriate voltage (i.e., outside the defined levels), the read will be indeterminate. Use of a port A pin for digital input does not preclude use as an analog input.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								RES10	STS		PRS				
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

ADCTL0 is used to select 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 in the MC68HC16Y1 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	0	0	0	0	0	0	0	0	0

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

SCAN — Scan Mode Selection Bit
- 0 = Single conversion sequence
- 1 = Continuous conversion

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

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

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

S8CM — Select Eight-Conversion Sequence Mode
- 0 = Four-conversion sequence
- 1 = Eight-conversion sequence

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

[CD:CA] — Channel Selection Field

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

The following table 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	RSLT0 – RSLT3
0	0	0	0	1	AN1	RSLT0 – RSLT3
0	0	0	1	0	AN2	RSLT0 – RSLT3
0	0	0	1	1	AN3	RSLT0 – RSLT3
0	0	1	0	0	AN4	RSLT0 – RSLT3
0	0	1	0	1	AN5	RSLT0 – RSLT3
0	0	1	1	0	AN6	RSLT0 – RSLT3
0	0	1	1	1	AN7	RSLT0 – RSLT3
0	1	0	0	0	RESERVED	RSLT0 – RSLT3
0	1	0	0	1	RESERVED	RSLT0 – RSLT3
0	1	0	1	0	RESERVED	RSLT0 – RSLT3
0	1	0	1	1	RESERVED	RSLT0 – RSLT3
0	1	1	0	0	V_{RH}	RSLT0 – RSLT3
0	1	1	0	1	V_{RL}	RSLT0 – RSLT3
0	1	1	1	0	$(V_{RH} - V_{RL}) / 2$	RSLT0 – RSLT3
0	1	1	1	1	TEST/RESERVED	RSLT0 – RSLT3
1	0	0	0	0	AN0	RSLT0 – RSLT7
1	0	0	0	1	AN1	RSLT0 – RSLT7
1	0	0	1	0	AN2	RSLT0 – RSLT7
1	0	0	1	1	AN3	RSLT0 – RSLT7
1	0	1	0	0	AN4	RSLT0 – RSLT7
1	0	1	0	1	AN5	RSLT0 – RSLT7
1	0	1	1	0	AN6	RSLT0 – RSLT7
1	0	1	1	1	AN7	RSLT0 – RSLT7
1	1	0	0	0	RESERVED	RSLT0 – RSLT7
1	1	0	0	1	RESERVED	RSLT0 – RSLT7
1	1	0	1	0	RESERVED	RSLT0 – RSLT7
1	1	0	1	1	RESERVED	RSLT0 – RSLT7
1	1	1	0	0	V_{RH}	RSLT0 – RSLT7
1	1	1	0	1	V_{RL}	RSLT0 – RSLT7
1	1	1	1	0	$(V_{RH} - V_{RL}) / 2$	RSLT0 – RSLT7
1	1	1	1	1	TEST/RESERVED	RSLT0 – RSLT7

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

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

ADSTAT — ADC Status Register **$YFF70E**

15	14	11	10	8	7	0
SCF	NOT USED		CCTR		CCF	
RESET:						
0	0 0 0 0	0	0 0 0	0	0 0 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 are used to store data after conversion is complete. Each register can be read from three different addresses in the register block. Data format depends on the address from which it is read.

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

Conversion result is unsigned right-justified data. Bits [9:0] are used for 10-bit 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). The figure below is a block diagram of the MCCI.

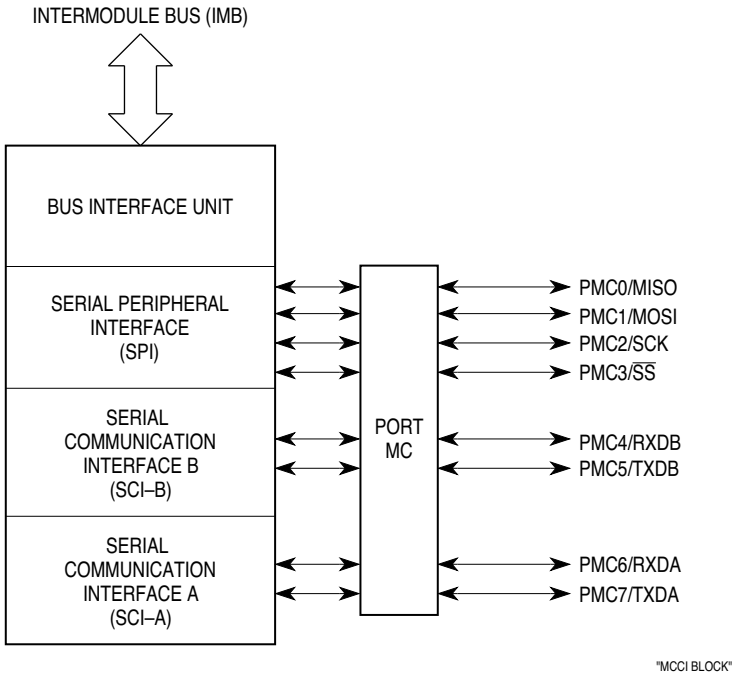


Figure 14 MCCI Block Diagram

The SCI provide standard nonreturn to zero (NRZ) mark/space format. Either will operate in full- or half-duplex mode — there are separate transmitter and receiver enable bits and dual data buffers for each interface. A modulus-type baud rate generator provides rates from 64 to 524 kbaud (with a 16.78-MHz system clock). Word length of either 8 or 9 bits is software selectable. Optional parity generation and detection provide either even or odd parity check capability. Advanced error detection circuitry catches glitches of up to 1/16 of a bit time in duration. Wakeup functions allow the CPU to run uninterrupted until meaningful data is available.

The SPI provides easy peripheral expansion or interprocessor communication via 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.

Table 29 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 (MIVR)
$YFFC06	SPI INTERRUPT REGISTER (ILSPI)	RESERVED
$YFFC08	RESERVED	MCCI PIN ASSIGNMENT (PMCPAR)
$YFFC0A	RESERVED	MCCI DATA DIRECTION (DDRMC)
$YFFC0C	RESERVED	MCCI PORT DATA REGISTER (PORTMC)
$YFFC0E	RESERVED	MCCI PORT PIN STATE (PORTMCP)
$YFFC10	RESERVED	
$YFFC12	RESERVED	
$YFFC14	RESERVED	
$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	RESERVED	
$YFFC22	RESERVED	
$YFFC24	RESERVED	
$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	RESERVED	
$YFFC32	RESERVED	
$YFFC34	RESERVED	
$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 state of the modmap bit in the SCIMCR. In the MC68HC16Y1, Y must equal $F — if M is cleared, IMB modules will be inaccessible until a reset occurs. M can be written only once after reset.

7.1 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 below. 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. Because the CPU16 in the MC68HC16Y1 drives only ADDR[19:0], ADDR[23:20] follow the logic state of ADDR19, and “Y” must equal $F — see the SCIM section of this summary for more information on how the state of MM affects the system.

7.1.1 MCCI Global Registers

Global registers contain parameters used by both the SPI and the SCI submodules. These parameters are used by the MCCI to interface with the CPU and other system modules.

MMCR — MCCI Configuration Register **$YFFC00**

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

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 may be negated by the CPU and by reset.

Bits [14:8] — Not Implemented

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 in the MC68HC16Y1 operates only in supervisor mode, SUPV has no meaning.

Bits [6:4] — Not Implemented

IARB — Interrupt Arbitration Identification Number

Each module that generates interrupts has an IARB field. The value in this field is used to arbitrate between simultaneous interrupt requests of the same priority. The reset value of all IARB fields other than that of the SCIM is $0 (lowest priority), to prevent priority conflict during initialization. The IARB field must be initialized to a value between $F (highest priority) and $1 (lowest priority), or subsequent interrupt requests will be identified by the CPU as spurious.

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	12	11	10	9	8	7						1	0
0	0	ILSCIB			ILSCIA			MIVR						1	1
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

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

ILSCIA, ILSCIB — Interrupt Level for SCIA, SCIB

ILSCIA, 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.

7.1.2 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 may alternately be assigned to 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	14	13	12	11	10	9	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	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								PMC7	PMC6	PMC5	PMC4	PMC3	PMC2	PMC1	PMC0

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	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								0	0	0	0	$\overline{SS}$	0	MOSI	MISO
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 actually used by the SPI submodule, and which pins are available for general-purpose I/O. 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.

$\overline{SS}$ — Slave Select

MOSI — Master Out Slave In

MISO — Master In Slave Out
0 = Pin is used for general-purpose I/O
1 = Pin is used by SPI

DDRMC — MCCI Data Direction Register **$YFFC0B**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								TXDA	RXDA	TXDB	RXDB	$\overline{SS}$	SCK	MOSI	MISO
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.

0 = Input
1 = Output

7.2 Serial Peripheral Interface

The SPI submodule communicates with external devices via 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.

7.2.1 SPI Pins

The SPI uses four bidirectional pins. These pins may be configured for general-purpose I/O when not needed for SPI application. The following table shows SPI pin functions

Table 30 SPI Pin Function

Pin Names	Mode	Function
Master In Slave Out (MISO)	Master Slave	Provides serial data input to the SPI Provides serial data 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

7.2.2 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 (SPSR). 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	6	5	4	3	2	1	0
SPIE	SPE	WOMP	MSTR	CPOL	CPHA	LSBF	SIZE	BAUD							
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 may 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

BAUD — Serial Clock 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 BR field. Giving BR a value of zero or one disables the baud rate generator. The following equations determine the SCK baud rate:

$$\text{SCK Baud Rate} = \text{System Clock}/(2 * \text{BR})$$

or

$$\text{BR} = \text{System Clock}/(2 * \text{SCK Baud Rate Desired})$$

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

7.2.3 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. Prior to 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 may 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 may 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.

7.3 Serial Communication Interface

There are two identical independent SCI systems, SCIA and SCIB, in the MCCI. Each 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.

7.3.1 SCI Pins

Two unidirectional transmit data pins, TXDA and TXDB, and two unidirectional receive data pins, RXDA and RXDB, are 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 shown below.

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

7.3.2 SCI Registers

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

All registers may 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 may 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
0	0	0	SCBR												
RESET:															
0	0	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.

Bits [15:13] — Not Implemented

SCBR — Baud Rate

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

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

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

where BR 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. In general, 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 prior to entering loop mode.

WOMS — Wired-OR Mode for SCI Pins

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

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

ILT — Idle-Line Detect Type

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

PT — Parity Type
0 = Even parity
1 = Odd parity
When parity is enabled, PT determines whether parity is even or odd for both the receiver and the transmitter.

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

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

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

M — Mode Select
0 = SCI frame: one start bit, eight data bits, one stop bit (ten bits total)
1 = SCI frame: one start bit, nine data bits, one stop bit (11 bits total)

WAKE — Wakeup by Address Mark
0 = SCI receiver awakened by idle-line detection
1 = SCI receiver awakened by address mark (last bit set)

TIE — Transmit Interrupt Enable
0 = SCI TDRE interrupts inhibited
1 = SCI TDRE interrupts enabled

TCIE — Transmit Complete Interrupt Enable
0 = SCI TC interrupts inhibited
1 = SCI TC interrupts enabled

RIE — Receiver Interrupt Enable
0 = SCI RDRF 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 may 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 may 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 commencing to send data.

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

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

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

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

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

TDRE — Transmit Data Register Empty Flag

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

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

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

TC — Transmit Complete Flag

0 = SCI transmitter is busy.

1 = SCI transmitter is idle.

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

RDRF — Receive Data Register Full Flag

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

1 = Register RDR contains new data.

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

RAF — Receiver Active Flag

0 = SCI receiver is idle.

1 = SCI receiver is busy.

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

IDLE — Idle-Line Detected Flag

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

1 = SCI receiver detected an idle-line condition.

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

OR — Overrun Error Flag

0 = RDRF is cleared before new data arrives.

1 = RDRF is not cleared before new data arrives.

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

NF — Noise Error Flag

0 = No noise detected on the received data.

1 = Noise occurred on the received data.

NF is set when the SCI receiver detects noise on a valid start bit, on any data bit, or on a stop bit. It is not set by noise on the idle line or on invalid start bits. Each bit is sampled three times. If 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	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	R8/T8	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0
RESET:															
0	0	0	0	0	0	0	U	U	U	U	U	U	U	U	U

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 it is configured for 8-bit operation, they have no meaning or effect.

8 Standby RAM with TPU Emulation

The standby RAM with TPU emulation module (TPURAM) contains a 2-Kbyte array of fast (two bus cycle) static RAM, which is especially useful for system stacks and variable storage. The RAM can be used to emulate TPU microcode ROM. The TPURAM can be mapped to any 2-Kbyte boundary in the address map, but must not overlap the module control registers — overlap makes the registers inaccessible. TPURAM responds to both program and data space accesses. Data can be read or written in bytes, word, or long words. The RAM is powered by V_{DD} in normal operation. During power-down, the RAM contents are maintained by power on standby voltage pin V_{STBY}. Power switching between sources is automatic.

8.1 TPURAM Register Block

TPURAM control registers occupy a 64-byte block. There are three TPURAM control registers in the block: the RAM module configuration register (TRAMMCR), the RAM test register (TRAMTST), and the RAM array base address register (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.

Table 31 TPURAM Control Register Address Map

Address	15 8	7 0
$YFFB00	RAM MODULE CONFIGURATION REGISTER (TRAMMCR)	
$YFFB02	RAM TEST REGISTER (TRAMTST)	
$YFFB04	RAM BASE ADDRESS AND STATUS REGISTER (TRAMBAR)	
$YFFB06–$YFFB3F	NOT IMPLEMENTED	

Y = M111, where M is the state of the modmap bit in the module configuration register of the single-chip integration module. In an MC68HC16Y1 system, M must always be set to one.

8.2 TPURAM Registers

Access to the TPURAM array is controlled by the RASP field in the TRAMMCR.

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

15			12				8	7 0
STOP			PDS				RASP	NOT USED
RESET:								
0			U				U	

Bits in TRAMMCR determine whether the RAM 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 RAM array can be powered by a standby power source (V_{STBY}) while V_{DD} to the microcontroller is turned off. PDS indicates when V_{STBY} has fallen below a reference 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.

RASP[1:0] — RAM Array Space Field

o = TPURAM array is placed in unrestricted space

1 = TPURAM array is placed in supervisor space.

This bit limits access to the SRAM array in microcontrollers that support separate user and supervisor operating modes. Because the CPU16 in the MC68HC16Y1 operates in supervisor mode only, RASP has no effect.

TRAMTST — RAM Test Register **$YFFB02**

TRAMTST is used for factory test of the TPURAM module.

TRAMBAR — RAM Base Address and Status Register **$YFFB04**

16												3		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	0	1

TRAMBAR is used to specify an array base address in the system memory map. This prevents accidental remapping of the array. TRAMBAR can be written only once after reset.

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

This field specifies bits [23:11] of the array base address. The array must be enabled in order to be accessed. Since the states of ADDR[23:20] follow the state of ADDR19 in the MC68HC16Y1, addresses in the range $080000 to $F7FFFF cannot be accessed.

RAMDS — RAM Array Disable Status Bit

0 = RAM array is enabled

1 = RAM array is disabled

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

8.3 TPURAM Operation

There are six TPURAM operating modes, as follows.

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

Standby mode is intended to preserve RAM contents when V_{DD} is removed. SRAM contents are maintained by V_{STBY}. Circuitry within the SRAM module switches to the higher of V_{DD} or V_{STBY} with no loss of data. When SRAM is powered by V_{STBY}, access to the array is not guaranteed.

Reset mode allows the CPU to complete the current bus cycle before resetting. When a synchronous reset occurs while a byte or word SRAM access is in progress, the access will be completed. If reset occurs during the first word access of a long-word operation, only the first word access will be completed. If reset occurs during the second word access of a long word operation, the entire access will be completed. Data being read from or written to the RAM may be corrupted by asynchronous reset.

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

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

The TPURAM array may be used to 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 via the IMB are inhibited and the control registers have no effect, allowing external RAM to emulate the TPURAM at the same addresses. See **4 Time Processor Unit** for more information.

9 Masked ROM Module

The masked ROM module (MRM) is designed to be used with the entire line of Motorola modular microcontrollers. The MRM consists of a fixed-location control register block and a memory array. Configuration information is contained in the register block. Default reset base address of the array in the system address map is specified by the customer, but the array may be remapped to other addresses. In addition to the array base address, the register block contains operating parameters, bootstrap code, and ROM verification information. An address map of the register block follows.

Table 32 MRM Control Register Address Map

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

Y = M111, where M is the state of the modmap bit in the module configuration register of the single-chip integration module. In an MC68HC16Y1 system, M must always be set to one.

The ROM array in the MC68HC16Y1 contains 48 Kbytes. It is arranged in 16-bit words, and is accessed via the intermodule bus. Bytes, words, and misaligned words can be accessed. Access time depends upon the number of wait states specified at mask programming time, but can be as fast as two system clocks for byte and aligned words. The MRM also responds to back-to-back IMB accesses to provide two bus cycle long word access.

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

The primary function of the MRM is to serve as nonvolatile memory for the microcontroller. It can be configured to support system bootstrap during reset. The CPU16 in the MC68HC16Y1 differentiates between program space accesses and data space accesses. The MRM array can be used for program code only, or for both program code and data. The MRM can also be programmed to insert wait states to accommodate migration from slower external development memory to the ROM array without retiming.

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

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

9.1 Masked ROM Control Registers

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

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

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

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

STOP — Stop Bit

0 = Normal ROM operation

1 = Disable ROM and activate emulator mode if enabled

Reset state of STOP is the complement of DATA14 state during reset. ROM array base address cannot be changed unless STOP is set.

BOOT — Boot ROM Control Bit

0 = CPU16 accesses ROM array addresses after reset

1 = CPU16 cannot access ROM array addresses after reset

Reset state of $\overline{BOOT}$ is specified by the user. Bootstrap function is overridden if STOP = 1.

LOCK — Lock Registers Bit

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

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

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

EMUL — Emulator Mode Control Bit

0 = Normal ROM operation

1 = MRM enters emulator mode when STOP is set.

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

ASPC — ROM Array Space Field

Because the MC68HC16Y1 operates only in supervisory mode, ASPC determines whether accesses are restricted solely to program space, or whether accesses are made to both program and data space. In systems with restricted access levels, ASPC also determines whether accesses are restricted solely to supervisor space. The reset state of ASPC is user specified. The table below shows ASPC encoding.

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

WAIT — Wait States Field

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

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

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

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

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

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

ROMBAH and ROMBAL are used to specify ROM array base address. They can only be written when STOP = 1 and LOCK = 0. This prevents accidental remapping of the array. Since the states of ADDR[23:20] follow the state of ADDR19 in the MC68HC16Y1, addresses in the range $080000 to $F7FFFF cannot be accessed. Because the 48 Kbyte ROM array in the MC68HC16Y1 must be mapped to a 64 Kbyte boundary, ROMBAL always contains $0000.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED													RSP18	RSP17	RSP16
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	USER SPECIFIED		

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

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

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

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

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

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

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

Typically, reset vectors for the system CPU are contained in nonvolatile memory and are only fetched when the CPU comes out of reset. The user can specify that these four words be used as reset vectors, and can specify the content of these locations. The content of these words cannot be changed. In the MC68HC16Y1, ROMBS0 to ROMBS3 correspond to system addresses $000000 to $000006.

10 Summary of Changes

This is a partial revision. Most of the publication remains the same, but the following changes were made to improve it. Typographical errors that do not affect content are not annotated.

Page 4	Block diagram revised. All pin functions shown, port mnemonics changed.
Pages 6–8	Corrected port assignments, new notes, changed B driver description.
Pages 9 & 11	Changed ADC analog input mnemonics and parallel port mnemonics to prevent confusion.
Page 15	Added XMSK, YMSK registers to diagram.
Page 36	SCIM address map standardized.
Page 39	Added RSR description.
Pages 56–59	New reset section.
Pages 59–61	New interrupts section.
Page 66	Corrected PORTFE reset state.
Page 77	Removed RSR from test register listing.
Page 79	TPU address map standardized.
Page 90	GPT address map standardized.
Pages 90 & 95	Changed GPT I/O port register mnemonics to reflect port name.
Page 102	ADC address map standardized.
Pages 102–109	Changed ADC analog input mnemonics and parallel port mnemonics to prevent confusion.
Page 106	Changed prescaler rate selection table to show %00000 setting is reserved.
Page 109	Added Result Register mnemonics.
Page 111	MCCI address map standardized.
Pages 111 & 114	Changed MCCI I/O port register mnemonics to reflect port name.

NOTES

How to reach us:
USA/EUROPE: Motorola Literature Distribution;
P.O. Box 20912; Phoenix, Arizona 85036. 1-800-441-2447
MFAX: RMFAX0@email.sps.mot.com - TOUCHTONE (602) 244-6609
INTERNET: http://Design-NET.com
JAPAN: Nippon Motorola Ltd.; Tatsumi-SPD-JLDC, Toshikatsu Otsuki,
6F Seibu-Butsuryu-Center, 3-14-2 Tatsumi Koto-Ku, Tokyo 135, Japan. 03-3521-8315
HONG KONG: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park,
51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298